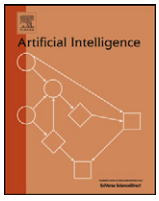
Artificial Intelligence 195 (2013) 1–43

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



A generalised framework for dispute derivations in assumption-based argumentation

Francesca Toni *

Imperial College London, South Kensington Campus, London SW72AZ, UK

ARTICLE INFO

Article history:
Received 8 July 2011
Received in revised form 19 September 2012
Accepted 25 September 2012
Available online 5 October 2012

Keywords:
Argumentation
Computation
Soundness and completeness

ABSTRACT

Assumption-based argumentation is a general-purpose argumentation framework with well-understood theoretical foundations and viable computational mechanisms (in the form of *dispute derivations*), as well as several applications. However, the existing computational mechanisms have several limitations, hindering their deployment in practice: (i) they are defined in terms of implicit parameters, that nonetheless need to be instantiated at implementation time; (ii) they are variations (for computing different semantics) of one another, but still require different implementation efforts; (iii) they reduce the problem of computing arguments to the problem of computing assumptions supporting these arguments, even though applications of argumentation require a justification of claims in terms of explicit arguments and attacks between them.

In this context, the contribution of this paper is two-fold. Firstly, we provide a unified view of the existing (GB-, AB- and IB-)dispute derivations (for computation under the grounded, admissible and ideal semantics, respectively), by obtaining them as special instances of a single notion of *X-dispute derivations* that, in addition, renders explicit the implicit parameters in the original dispute derivations. Thus, X-dispute derivations address issues (i) and (ii). Secondly, we define *structured X-dispute derivations*, extending X-dispute derivations by computing explicitly the underlying arguments and attacks, in addition to assumptions. Thus, structured X-dispute derivations also address issue (iii). We prove soundness and completeness results for appropriate instances of (structured) X-dispute derivations, w.r.t. the grounded, admissible and ideal semantics, thus laying the necessary theoretical foundations for deployability thereof.

1. Introduction

Assumption-based argumentation (ABA) [3,15] is a general-purpose argumentation framework with a wide range of applications in default reasoning [3], legal reasoning [37], decision-making [17,38] and medicine [9]. ABA admits several instances, including many existing non-monotonic logics [3,10] with, in particular, logic programming with negation as failure. ABA (and its instances) can be equipped with several argumentation semantics, all determining a notion of "acceptability" of sets of assumptions. Indeed, ABA equates the problem of determining "acceptable" arguments to the problem of determining "acceptable" sets of assumptions supporting these arguments. Arguments are deductions of claims using (inference) rules and are supported by sets of assumptions, and attacks are directed at the assumptions in the support of arguments, in that they are arguments for the contrary of (one of these) assumptions.

* Tel.: +44 (0) 20 75948228; fax: +44 (0) 20 75948932.
E-mail address: ft@imperial.ac.uk.

0004-3702/$ – see front matter
http://dx.doi.org/10.1016/j.artint.2012.09.010

The argumentation semantics for sanctioning "acceptability" include the admissible semantics [3,14], the grounded semantics [3,14], and the ideal semantics [14]. Under any of these semantics, ABA is an instance of abstract argumentation [12], in that any ABA framework can be naturally mapped onto an abstract argumentation framework and the "acceptable" sets of arguments (under the admissible [12], grounded [12] or ideal [14] semantics) for this derived framework are in one-to-one correspondence with the "acceptable" sets of assumptions (under the ABA notions of admissible, grounded or ideal semantics) for the original ABA framework (see [14]). ABA is thus a concrete (non-abstract) argumentation framework but with well-understood relationships with abstract argumentation. Furthermore, it fulfils properties of rationality [43] and is well suited for practical reasoning [18]. Finally, the computational complexity of various reasoning tasks in several instances of ABA has been studied [10,20].

In addition to well-understood theoretical foundations and applications, a number of computational mechanisms have been defined for ABA, for computing admissible, grounded and ideal supports for conclusions/claims [13,14,29,25,26]. Here we focus on the *AB-, GB- and IB-dispute derivations* of [13,14]. These can be seen as games for conducting disputes amongst two fictional players, the proponent of a claim and some opponent, both using argumentation. Dispute derivations allow to determine whether the claim put forward by the proponent is supported by a set of arguments that can be deemed to be "acceptable" under the admissible, grounded and ideal semantics respectively [3,14]. The disputes are defined as sequences of tuples, each representing a dispute step. Each tuple holds information about (some of) the assumptions made so far in the dispute to support arguments by the proponent and the opponent, and the currently pending issues in the dispute. The disputes manipulate several (multi-)sets of assumptions, held in the tuples, rather than arguments, thus allowing to guarantee the computation of arguments that are (semantically) relevant to the claim at stake. The three kinds of dispute derivations differ in their use of different "filtering mechanisms" and in their use of data structures, given to model disputes conducted according to different semantics.

These existing computational mechanisms have several limitations:

(i) They are defined in terms of implicit parameters (for example to decide which player should play next), that nonetheless need to be instantiated at implementation time and whose instantiation affects the resulting system.
(ii) They are variations (for computing the different semantics) of one another, but still require different implementation efforts.
(iii) They hide the arguments and attacks between them, implicitly manipulated during disputes. In particular, AB-, GB- and IB-dispute derivations explore implicitly a dialectical structure of arguments by the proponent, counter-arguments by the opponent, arguments by the proponent attacking the counter-arguments and so on. However, while doing so, dispute derivations only keep track of (some of) the assumptions underlying these arguments, and the dialectical structure is lost.

The first two issues render the implementation of dispute derivations an arduous task, and determining the correctness of the implementation w.r.t. the computational mechanism difficult. The third issue hinders the usefulness of dispute derivations, since applications of argumentation in general and ABA in particular (for example for medical-decision support [31,9]) tend to require a justifications of the acceptability of claims in terms of explicit arguments and attacks between them.

This paper deals with these issues as follows. Firstly, we propose a notion of *X-dispute derivation*, whose parameters can be suitably instantiated to obtain each of AB-, GB- and IB-dispute derivations. X-dispute derivations can be thus seen as a unified framework for presenting and understanding dispute derivations for ABA. Moreover, X-dispute derivations allow to understand, at a formal high-level view, different design choices underlying the various kinds of existing dispute derivations. Finally, X-dispute derivations render explicit choices that any implementation of AB-, GB- and IB-dispute derivations need to make, thus paving the way towards a novel, unified and modular implementation. Thus, X-dispute derivations address issues (i) and (ii) above. Secondly, in order to address issue (iii), we define *structured X-dispute derivations*, an extension of X-dispute derivations computing explicitly the dialectical structure hidden in X-dispute derivations. Thus, structured X-dispute derivations provide a hybrid ABA-abstract argumentation mechanism, exploiting and demonstrating the correspondence, given in [14], between "acceptable" supports for conclusions, in terms of sets of assumptions, and "acceptable" sets of arguments for these conclusions, in the abstract sense of [12], for all notions of "acceptability" considered above.

Our structured X-dispute derivations are in the spirit of the structured AB-dispute derivations of [26], computing the admissible semantics, but are defined parametrically, in the spirit of X-dispute derivations, and can be instantiated for computing the grounded and ideal semantics as well as the admissible semantics.

We study soundness and completeness of X-dispute derivations and structured X-dispute derivations, building upon the correspondence between X-dispute derivations and AB-, GB- and IB-dispute derivations and between structured X-dispute derivations and X-dispute derivations. The results clearly characterise, in addition to the "acceptability" of computed sets of assumptions, also (for structured X-dispute derivations) the "acceptability" of the computed dialectical structures, by linking them with the "acceptable" dispute trees of [13,14]. We also consider the relationships between (instances of) X-dispute derivations and proof procedures for negation as failure in logic programming.

The paper is organised as follows. In Section 2 we give and illustrate background on ABA. In Section 3 we summarise and illustrate AB-, GB- and IB-dispute derivations, and single out explicitly choices underlying their definitions (but implicit in [13,14]). In Section 4 we define X-dispute derivations. In Section 5 we give several properties of X-dispute derivations, including showing how they generalise AB-, GB- and IB-dispute derivations and how specific instances of X-dispute deriva-

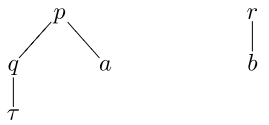


Fig. 1. Examples of arguments for Example 2.1: argument a for p (left) and b for r (right).

tions in the instance of ABA for logic programming relate to SLDNF and the abductive proof procedure of [23]. In Section 6 we define structured X-dispute derivations. In Section 7 we show their soundness w.r.t. admissible, grounded and ideal semantics, by proving, in particular, that structured X-dispute derivations and X-dispute derivations are in one-to-one correspondence. In Section 8 we give completeness results for X-dispute derivations as well as structured X-dispute derivations. In Section 9 we consider soundness and completeness of structured X-dispute derivations w.r.t. argumentation semantics other than the admissible, grounded and ideal semantics. In Section 10 we discuss related work. In Section 11 we conclude.

2. Assumption-based argumentation

This section provides the basic background on assumption-based argumentation (ABA), see [3,13,14] for additional details.

An *ABA framework* is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ where

- $(\mathcal{L}, \mathcal{R})$ is a *deductive system*, consisting of a language $\mathcal{L}$ and a set $\mathcal{R}$ of inference rules;
- $\mathcal{A} \subseteq \mathcal{L}$, referred to as the set of *assumptions*;
- $\bar{}$ is a (total) mapping from $\mathcal{A}$ into $\mathcal{L}$, where $\bar{\alpha}$ is referred to as the *contrary* of α.

We will assume that the inference rules in $\mathcal{R}$ have the syntax $\sigma_0 \leftarrow \sigma_1, \ldots, \sigma_n$ (for $n \geqslant 0$) where $\sigma_i \in \mathcal{L}$. We will refer to σ_0 and $\sigma_1, \ldots, \sigma_n$ as the *head* and the *body* of the rule, respectively. We will sometimes represent $\sigma_0 \leftarrow$ simply as σ_0. As in [13], we will restrict attention to *flat* ABA frameworks, such that if $\alpha \in \mathcal{A}$, then there exists no inference rule of the form $\alpha \leftarrow \sigma_1, \ldots, \sigma_n \in \mathcal{R}$, for any $n \geqslant 0$. Logic programming and default logic are examples of flat ABA frameworks (see [3]).

An *argument* for a sentence $\sigma \in \mathcal{L}$ supported by a set of assumptions A is a defeasible proof of σ from A, obtained by applying backwards the rules in $\mathcal{R}$ until only assumptions are left. This defeasible proof can be in the form of a tree [15] or a backward deduction [13] or a forward deduction [3]. Here we consider the former.

Definition 2.1. (See [15].) A *proof* for $\sigma \in \mathcal{L}$ supported by $S \subseteq \mathcal{L}$ is a (finite) tree with nodes labelled by sentences in $\mathcal{L}$ or by τ,[1] such that

- the root is labelled by σ;
- for every node N:
  - if N is not a leaf and σ_N is the label of N, then there is an inference rule $\sigma_N \leftarrow \sigma_1, \ldots, \sigma_m$ ($m \geqslant 0$) and either $m = 0$ and the child of N is τ or $m > 0$ and N has m children, labelled by $\sigma_1, \ldots, \sigma_m$ (respectively);
- S is the set of all sentences in $\mathcal{L}$ labelling the leaves.

An *argument* for $\sigma \in \mathcal{L}$ supported by a set of assumptions $A \subseteq \mathcal{A}$ is a proof for σ supported by A.

Example 2.1. Given an ABA framework with[2]

- $\mathcal{R} = \{p \leftarrow q, a;\ q \leftarrow;\ r \leftarrow b\}$;
- $\mathcal{A} = \{a, b\}$;
- $\bar{a} = r$, $\bar{b} = s$.

Fig. 1 shows an argument a for p supported by $\{a\}$ and an argument b for r supported by $\{b\}$.

In order to determine whether a conclusion (sentence) should be drawn, a set of assumptions needs to be identified providing an "acceptable" support for the conclusion. Various notions of "acceptable" support can be formalised, using a notion of "attack" amongst sets of assumptions whereby *A attacks B* iff there is an argument for some $\bar{\alpha}$ supported by

[1] The symbol τ intuitively stands for "true" and is such that $\tau \notin \mathcal{L}$. It allows to distinguish between facts, namely inference rules with an empty set of premises, and assumptions.

[2] Here and in all examples in the paper, for simplicity, we omit to give explicitly the $\mathcal{L}$ component of the ABA framework. It is intended, implicitly, to be the set of all sentences occurring in the other components.

(a subset of) A where α is in B (and thus, for instance, $\{b\}$ attacks $\{a\}$ in Example 2.1). Then, a set of assumptions is deemed

- *admissible*, iff it does not attack itself and it attacks every set of assumptions attacking it [3];
- *preferred*, iff it is maximally (w.r.t. set inclusion) admissible [3];
- *grounded*, iff it is minimally (w.r.t. set inclusion) complete, where a set of assumptions is *complete* iff it is admissible and it contains all assumptions it can defend, by counter-attacking all attacks against them [3];
- *ideal*, iff it is admissible and contained in all preferred sets [14].

All these notions are possible formalisations of the notion of "acceptable" support for a conclusion. The first two are *credulous* notions, possibly sanctioning several alternative sets as "acceptable" supports, and the third is a *sceptical* notion, always sanctioning one single set as "acceptable" support. The last notion can be used as a sceptical notion by considering the maximal (w.r.t. set inclusion) ideal set, as this is also unique [14]. We will mostly focus on admissible, grounded and ideal sets of assumptions.

As shown in [14], a correspondence exists between "acceptable" supports for conclusions, in terms of sets of assumptions as given above, and "acceptable" sets of arguments for these conclusions, in the abstract sense of [12], for all notions of "acceptability" given above, as follows. Given an abstract argumentation framework (namely a set of arguments and a binary attack relation between arguments in the given set), for sets of arguments $\mathtt{A}$ and $\mathtt{B}$ (subsets of the given set), we say that $\mathtt{A}$ attacks $\mathtt{B}$ iff some argument in $\mathtt{A}$ attacks some argument in $\mathtt{B}$. Then, a set of arguments is

- *admissible*, iff it does not attack itself and it attacks every set of arguments attacking it [12];
- *preferred*, iff it is maximally (w.r.t. set inclusion) admissible [12];
- *grounded*, iff it is minimally (w.r.t. set inclusion) complete, where a set of arguments is *complete* iff it is admissible and it contains all arguments it can defend, by counter-attacking all attacks against them [12];
- *ideal*, iff it is admissible and contained in all preferred sets [14].

Then, given that an ABA-argument attacks another if the former supports the contrary of an assumption in the support of the latter, the correspondence between the assumption-view and the argument-view of ABA can be summarised as follows [14]:

- if a set of assumptions A is admissible/grounded/ideal then the union of all arguments supported by any subset of A is admissible/grounded/ideal;
- if a set of arguments $\mathtt{A}$ is admissible/grounded/ideal then the union of all sets of assumptions supporting the arguments in $\mathtt{A}$ is admissible/grounded/ideal.

Three kinds of *dispute trees* can be introduced in correspondence with admissible/grounded/ideal sets of arguments, as in [14,15]. Formally, a dispute tree for an argument $\mathtt{a}$ is a (possibly infinite) tree $\mathcal{T}$ such that

1. every node of $\mathcal{T}$ is labelled by an argument and is assigned the status of *proponent* node or *opponent* node, but not both;
2. the root is a proponent node labelled by $\mathtt{a}$;
3. for every proponent node N labelled by an argument $\mathtt{b}$, and for every argument $\mathtt{c}$ that attacks $\mathtt{b}$, there exists a child of N, which is an opponent node labelled by $\mathtt{c}$;
4. for every opponent node N labelled by an argument $\mathtt{b}$, there exists exactly one child of N which is a proponent node labelled by an argument which attacks $\mathtt{b}$;
5. there are no other nodes in $\mathcal{T}$ except those given by 1–4 above.

Then, a dispute tree is

- *admissible* iff no argument labels both a proponent and an opponent node;
- *grounded* iff it is finite;
- *ideal* iff it is admissible and for no opponent node O in it there exists an admissible dispute tree for the argument labelling O.

The following example illustrates the notions of grounded, admissible and ideal dispute trees.

Example 2.2. Given an ABA framework with

- $\mathcal{R} = \{p \leftarrow q, a; q \leftarrow; r \leftarrow b; s \leftarrow c; s \leftarrow a\}$,
- $\mathcal{A} = \{a, b, c\}$,
- $\bar{a} = r$, $\bar{b} = s$, $\bar{c} = t$,

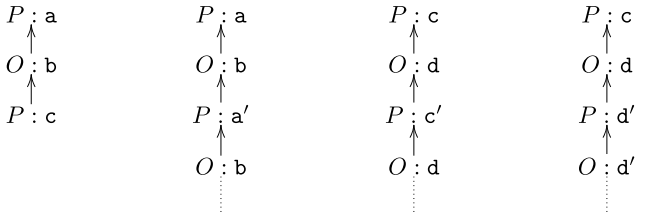


Fig. 2. Dispute trees for Example 2.2.

let a and b be the arguments in Fig. 1, c be the argument for s supported by $\{c\}$ and a′ be the argument for s supported by $\{a\}$. Then, the left-most tree in Fig. 2 is a grounded, admissible and ideal dispute tree, the second tree (with $P : \mathtt{a}'$ child of $O : \mathtt{b}$ ad infinitum) is an admissible and ideal dispute tree, but not a grounded dispute tree. If we add $t \leftarrow d$ to $\mathcal{R}$, d to $\mathcal{A}$, and set $\overline{d} = c$, then, for arguments d for t supported by $\{d\}$ and c′ for c supported by $\{c\}$, the third tree in Fig. 2 is an admissible dispute tree, but not a grounded (as it is infinite) or ideal (as there is an admissible dispute tree for d) dispute tree. Finally, if we set $\overline{d} = d$ instead, then, for argument d′ for d supported by $\{d\}$, the fourth (right-most) tree in Fig. 2 is a dispute tree, but not admissible or grounded or ideal.

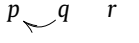
Let us refer to the set of all arguments belonging to the proponent nodes in a dispute tree $\mathcal{T}$ as the *argument defence set* of $\mathcal{T}$. Then, following [14] and [15]:

- the argument defence set of an admissible dispute tree is admissible;
- the argument defence set of a grounded/ideal dispute tree is a subset of the grounded/ideal (respectively) set of arguments;
- if an argument a belongs to an admissible/grounded/ideal set of arguments A then there exists an admissible/grounded/ideal (respectively) dispute tree for a with argument defence set A′ such that $\mathtt{A}' \subseteq \mathtt{A}$ and A′ is admissible.[3]

Finally, some of our results will be given for p-acyclic ABA frameworks, as defined in [14] and reviewed below. Given an ABA framework $\mathcal{AF}$, $\mathcal{AF}^+$ denotes the ABA framework obtained by deleting all assumptions appearing in the premises of the rules in $\mathcal{AF}$. The *dependency graph of* $\mathcal{AF}^+$ is a directed graph where:

- the nodes are the atoms occurring in $\mathcal{AF}^+$;
- a (directed) arc from a node σ to a node σ' is in the graph iff there exists a rule $\sigma \leftarrow \sigma_1, \ldots, \sigma_n$ in $\mathcal{AF}^+$ such that $\sigma' = \sigma_i$ for some $i = 1, \ldots, n$.

Then, $\mathcal{AF}$ is *p-acyclic* if the dependency graph of $\mathcal{AF}^+$ is acyclic. As an example, for $\mathcal{AF}$ in Example 2.1, $\mathcal{AF}^+$ has rules $p \leftarrow q; q \leftarrow; r \leftarrow$ with dependency graph



Since this graph is acyclic, $\mathcal{AF}$ is p-acyclic. As another example, given some $\mathcal{AF}$ with rule $p \leftarrow p$ (where $p \notin \mathcal{A}$), the dependency graph of $\mathcal{AF}^+$ has a cycle (from p to itself) and $\mathcal{AF}$ is not p-acyclic. Intuitively, arguments can be computed finitely, top-down, in p-acyclic ABA frameworks. Since GB-, AB-, IB-dispute derivations (implicitly) incorporate the computation of arguments top-down, p-acyclicity is an important condition to guarantee completeness of these dispute derivations (see [14]), which we review in the next section.

3. GB-, AB-, IB-dispute derivations

This section summarises and illustrates GB-, AB- and IB-dispute derivations [13,14]. Their formal definition is reported in Appendix A.

At a high level of abstraction, each of GB-, AB- and IB-dispute derivations can be understood as a game between two (fictional) players – a *proponent* and an *opponent* – with rules roughly as follows: the opponent can dispute an argument of the proponent by attacking one of the argument's supporting assumptions; the proponent can in turn defend its arguments by counter-attacking the opponent's attacks with other arguments, possibly with the aid of other defending assumptions; the proponent cannot attack any of its own assumptions. The game can have a successful outcome, and return an "acceptable" (admissible, grounded, ideal, respectively) set of assumptions supporting and defending the given claim, or fail to provide such an outcome, if the claim cannot be defended. This computational model incorporates several *filtering* mechanisms (but different kinds for AB-, GB-, and IB-dispute derivations), allowing to avoid re-computation and using a storing mechanism

[3] In the grounded case we also need to assume that the set of all arguments for the given ABA framework is finite, see [15], in order to avoid grounded sets of arguments requiring trees infinite in breadth. Note that, if $\mathcal{L}$ is finite, then the set of all arguments is guaranteed to be finite too.

for assumptions that have already been encountered earlier in the computation and defended (if held by the proponent) or defeated (if held by the opponent). The first kind of assumptions is referred to as *defences* and the second as *culprits*.

Formally, AB- and GB-dispute derivation are sequences of tuples of the form:

$$\langle \mathcal{P}, \mathcal{O}, D, C \rangle$$

and IB-dispute derivations are sequences of tuples of the form:

$$\langle \mathcal{P}, \mathcal{O}, D, C, \mathcal{F} \rangle$$

whose components hold the (assumptions underlying some of the) arguments by the proponent ($\mathcal{P}$) and opponent ($\mathcal{O}$), defences (D) and culprits (C), and a set of (assumptions supporting) arguments by the opponent ($\mathcal{F}$) that need to be checked (in the case of IB-dispute derivations) using a *Fail* predicate defined, as in [14], as follows.

Definition 3.1. (See [14].) Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ be an ABA framework and $S \subseteq \mathcal{L}$.[4] *Fail*(S) holds iff there exists no admissible $A \subseteq \mathcal{A}$ such that, for each $\sigma \in S$, there exists an argument for σ supported by some A' with $A' \subseteq A$.[5]

We illustrate the three notions of dispute derivations by means of examples.

Example 3.1. Consider the ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ with

- $\mathcal{R} = \{p \leftarrow a; q \leftarrow b; r \leftarrow c\}$;
- $\mathcal{A} = \{a, b, c\}$;
- $\bar{a} = q$, $\bar{b} = r$, $\bar{c} = s$.

The following is a GB-dispute derivation of $\{a, c\}$ for p:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C
0	$\{p\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{a\}$	$\{\}$	$\{a\}$	$\{\}$
2	$\{\}$	$\{\{q\}\}$	$\{a\}$	$\{\}$
3	$\{\}$	$\{\{b\}\}$	$\{a\}$	$\{\}$
4	$\{r\}$	$\{\}$	$\{a\}$	$\{b\}$
5	$\{c\}$	$\{\}$	$\{a, c\}$	$\{b\}$
6	$\{\}$	$\{\{s\}\}$	$\{a, c\}$	$\{b\}$
7	$\{\}$	$\{\}$	$\{a, c\}$	$\{b\}$

At step 1 the proponent ($\mathcal{P}$) has completed the construction of an argument for p supported by $\{a\}$, and a has been recorded as a defence (in D). At step 3 the opponent ($\mathcal{O}$) has completed the construction of all arguments against the (argument by) the proponent: these amount to a single argument in this case, for q (the contrary of a) and supported by $\{b\}$. At step 4 the proponent chooses the culprit b in the support of this argument (and b is added to C) and starts building a counter-attack against it. This counter-attack, in the form of an argument for r (the contrary of b) supported by $\{c\}$, is completed at step 5 (when c is also added to D). The opponent's attempt to build attacks against this new argument by the proponent fails (steps 6 and 7) and the GB-dispute derivation succeeds.

Consider the same ABA framework but with $\mathcal{R}$ replaced by $\mathcal{R}' = \{p \leftarrow a; q \leftarrow b, a; r \leftarrow c; r \leftarrow b\}$. Then, the earlier GB-dispute derivation with $\mathcal{O}$ at step 3 replaced by $\{\{b, a\}\}$ is still a GB-dispute derivation of $\{a, c\}$ for p. Note that this time however the following steps of *filtering* may be performed (depending on the implementation choices underlying the construction of the derivation, as we will discuss later):

- *filtering of culprits by defences*: at step 3, a cannot be selected as a culprit in $\mathcal{O}$, as $a \in D$, namely it has already been chosen as a defence (see case 2(i)(b) in Definition A.1 in Appendix A);
- *filtering of defences by culprits*: at step 5, $r \leftarrow b$ cannot be chosen to construct an argument for r, namely b cannot be chosen as a defence, as $b \in C$, namely it has already been chosen as a culprit (see case 1(ii) in Definition A.1 in Appendix A).

GB-dispute derivations incorporate these two kinds of filtering in order to guarantee that the computed defences are not self-attacking, achieved by enforcing that the same assumption cannot be both a defence and a culprit. Besides filtering of

[4] Following [13,14], in this and all definitions that follow sets are actually multi-sets, but we use the same symbols for multi-set membership, union, intersection, and power set as for ordinary sets.

[5] In [14], a notion of *Fail-dispute derivation* is given to determine whether *Fail*(S) holds for any input S. In this paper we ignore the computation of *Fail*, as the same notion of Fail-dispute derivation as in [14] can be deployed.

culprits by defences and filtering of defences by culprits, AB- and IB-dispute derivations incorporate two additional forms of filtering, given below.

Example 3.2. Consider the ABA framework at the beginning of Example 3.1 with $\mathcal{R}$ extended to also include $s \leftarrow b$. Steps 0–6 of the GB-dispute derivation form the beginning of an AB-dispute derivation of a defence set $\{a, c\}$ for p, concluded by the following steps

Step	$\mathcal{P}$	$\mathcal{O}$	D	C
$7'$	{}	$\{\{b\}\}$	$\{a, c\}$	$\{b\}$
8	{}	{}	$\{a, c\}$	$\{b\}$

At step 8, *filtering of culprits by culprits* has been performed, by dropping $\{b\}$ from $\mathcal{O}$ since $b \in C$, to avoid the re-computation of a counter-attack for a culprit (b) that has already been dealt with.

The following AB-dispute derivation of $\{c\}$ for r shows the use of *filtering of defences by defences* (at step 5), to avoid the re-computation of a defence that has already been dealt with:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C
0	$\{r\}$	{}	{}	{}
1	$\{c\}$	{}	$\{c\}$	{}
2	{}	$\{s\}$	$\{c\}$	{}
3	{}	$\{\{b\}\}$	$\{c\}$	{}
4	$\{r\}$	{}	$\{c\}$	$\{b\}$
5	{}	{}	$\{c\}$	$\{b\}$

At step 5, the support c for r is filtered out from $\mathcal{P}$ since it is already in D.

AB-dispute derivations incorporate these two kinds of filtering in order to finitely compute infinite admissible dispute trees (see [13]). IB-dispute derivations incorporate these two additional kinds of filtering, as well as a *Fail* check (see Definition 3.1).

Example 3.3. Consider $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ with

- $\mathcal{R} = \{\neg a \leftarrow a; \neg a \leftarrow b; \neg b \leftarrow a; \neg c \leftarrow d; \neg d \leftarrow c\}$;
- $\mathcal{A} = \{a, b, c, d\}$;
- $\overline{\alpha} = \neg\alpha$ for all $\alpha \in \mathcal{A}$.

The following is an IB-dispute derivation of $\{b\}$ for $\neg a$:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$
0	$\{\neg a\}$	{}	{}	{}	{}
1	$\{b\}$	{}	$\{b\}$	{}	{}
2	{}	$\{\{\neg b\}\}$	$\{b\}$	{}	{}
3	{}	$\{\{a\}\}$	$\{b\}$	{}	{}
4	$\{\neg a\}$	{}	$\{b\}$	$\{a\}$	$\{\{a\}\}$
5	{}	{}	$\{b\}$	$\{a\}$	$\{\{a\}\}$
6	{}	{}	$\{b\}$	$\{a\}$	{}

At step 4, the support $\{a\}$ of a completed argument by the opponent is "moved" to the $\mathcal{F}$ component. At step 6, $\mathcal{F}$ is selected and "emptied", since $\mathit{Fail}(\{a\})$ holds.

IB-dispute derivations also rely upon a marking mechanism, summarised (and deployed) in Section 4.

The various kinds of dispute derivations vary in the form of filtering they deploy and on whether they use the $\mathcal{F}$ component. Also, a number of choices need to be made by any implementation of these mechanisms, as discussed in [25]:

- a selection function for choosing sentences in $\mathcal{P}$ or in elements of $\mathcal{O}$; an example (for $\mathcal{O}$) arises in the modification with $\mathcal{R}'$ in Example 3.1: at the modified step 3, $\mathcal{O} = \{\{b, a\}\}$ and the selection function needs to decide which of b or a will be considered first as a possible culprit;
- a mechanism for choosing an element in $\mathcal{O}$; an example would arise if an additional rule $q \leftarrow d$, with $d \in \mathcal{A}$, were added to $\mathcal{R}$ in Example 3.1: at step 3, $\mathcal{O}$ would be $\{\{b\}, \{d\}\}$ and this mechanism would decide which of $\{b\}$ or $\{d\}$ would be considered first;

- a mechanism for choosing an element in $\mathcal{F}$; an example would arise if an additional rule $\neg b \leftarrow d$ were added to $\mathcal{R}$ in Example 3.3: at step 3, $\mathcal{O}$ would be $\{\{a\}, \{d\}\}$ and, at step 4, $\mathcal{F}$ would be $\{\{a\}, \{d\}\}$, and this mechanism would decide which of $\{a\}$ or $\{d\}$ would be considered first;
- a mechanism for deciding which activity to perform amongst operating on the $\mathcal{P}$, $\mathcal{O}$ or $\mathcal{F}$ elements of a tuple; in Example 3.3, at step 5 the decision was to operate on $\mathcal{P}$, rather than $\mathcal{F}$.

With the exception of the selection function, these choices (and the corresponding mechanisms) are implicit in the definition of GB-, AB- and IB-dispute derivations.

In Section 4 we will give a new notion of *X-dispute derivations* generalising all of the existing GB-, AB- and IB-dispute derivations, and rendering all parameters (filtering, use of $\mathcal{F}$ and choices for the implementation) explicit.

The construction of arguments and attacks between them is also implicit in GB-, AB- and IB-dispute derivations. For instance, the (first) GB-dispute derivation of Example 3.1 implicitly constructs arguments

- a for p supported by $\{a\}$,
- b for q supported by $\{b\}$,
- c for r supported by $\{c\}$

such that c attacks b and b attacks a. In Section 6, we will give a new notion of *structured X-dispute derivations* generalising X-dispute derivations and rendering arguments and attacks explicit.

4. X-dispute derivations

X-dispute derivations are defined, like IB-dispute derivations, as sequences of tuples of the form $\langle \mathcal{P}, \mathcal{O}, D, C, \mathcal{F} \rangle$ but in terms of a number of parameters for:

1. filtering
2. deciding how the $\mathcal{F}$ component should be updated
3. explicit choices that any implementation of X-dispute derivations needs to make

We will see, in Section 5.1, that GB-, AB- and IB-dispute derivations can be obtained as instances of X-dispute derivations for specific instances of the first two kinds of parameters. We will also see, in Section 5.2, that two existing proof procedures for logic programming can be obtained as instances of X-dispute derivations for specific instances of all kinds of parameters. In this section, we define these parameters abstractly.

Definition 4.1. The *filtering mechanisms* are:

- $f_{DbyC} : \wp(\mathcal{L}) \times \wp(\mathcal{L}) \mapsto \{true, false\}$;
  given $R, C \subseteq \mathcal{L}$, $f_{DbyC}(R, C)$ is referred to as (the outcome of) *filtering of defences* (R) *by culprits* (C);
- $f_{DbyD} : \wp(\mathcal{L}) \times \wp(\mathcal{L}) \mapsto \wp(\mathcal{L})$;
  given $R, D \subseteq \mathcal{L}$, $f_{DbyD}(R, D)$ is referred to as (the outcome of) *filtering of defences* (R) *by defences* (D);
- $f_{CbyD} : \mathcal{L} \times \wp(\mathcal{L}) \mapsto \{true, false\}$;
  given $\sigma \in \mathcal{L}, D \subseteq \mathcal{L}$, $f_{CbyD}(\sigma, D)$ is referred to as (the outcome of) *filtering of culprits* (σ) *by defences* (D);
- $f_{CbyC} : \wp(\mathcal{L}) \times \wp(\mathcal{L}) \mapsto \{true, false\}$;
  given $S, C \subseteq \mathcal{L}$, $f_{CbyC}(S, C)$ is referred to as (the outcome of) *filtering of culprits* (S) *by culprits* (C).

As we will see in Sections 5, our results for all instances of X-dispute derivations we will consider in this paper $f_{DbyC}(R, C) = true$ iff R and C have no elements in common, and $f_{CbyD}(\sigma, D) = true$ iff σ belongs to D. Moreover, all instances of f_{CbyC} and f_{DbyD} we will consider will be such that $f_{DbyD}(R, C)$ is contained in R and $f_{CbyC}(S, C) = true$ only if S and C have some elements in common. We refer to choices of the filtering mechanisms that meet these constraints as *canonical*, formally defined as follows:

Definition 4.2. The *filtering mechanisms* are said to be *canonical* if they fulfil the following properties:

- $f_{DbyC}(R, C) = (R \cap C = \{\})$;
- $f_{CbyD}(\sigma, D) = (\sigma \notin D)$;
- $f_{DbyD}(R, D) \subseteq R$;
- if $f_{CbyC}(S, C) = true$ then $S \cap C \neq \{\}$.

Throughout the paper, unless specified otherwise, we will leave the filtering mechanisms completely generic.

The parameter for deciding how $\mathcal{F}$ should be updated can be abstractly defined as follows:

Definition 4.3. The *update parameter* is:

- $updt : \wp(\wp(\mathcal{A})) \times \wp(\wp(\mathcal{A})) \mapsto \wp(\wp(\mathcal{A}))$.

Given $\mathcal{F}, S \subseteq \wp(\mathcal{A})$, $updt(\mathcal{F}, S)$ is referred to as the *S-update of* $\mathcal{F}$.

The implementation choice parameters can be abstractly defined as follows:

Definition 4.4. The *implementation choice parameters* are:

- $sel : \wp(\mathcal{L}) \mapsto \mathcal{L}$, referred to as *selection function*;
- $member\mathcal{O} : \wp(\wp(\mathcal{L})) \mapsto \wp(\mathcal{L})$;
- $member\mathcal{F} : \wp(\wp(\mathcal{L})) \mapsto \wp(\mathcal{L})$;
- $turn : \mathbb{N} \mapsto \{\mathcal{P}, \mathcal{O}, \mathcal{F}\}$.

Note that, with an abuse of notation, for simplicity, *turn* is used as a function of the step i of a derivation rather than as a function of the $\mathcal{P}, \mathcal{O}, \mathcal{F}$ components of the tuple at step i.[6]

Like for filtering mechanisms, we can (and will) make *canonical* choices for these parameters:

Definition 4.5. The *update and implementation choice parameters* are said to be *canonical* if they fulfil the following properties:

- $updt(\mathcal{F}, S) \supseteq \mathcal{F}$;
- if $S \neq \{\}$ then $sel(S) \in S$;
- if $\mathcal{O} \neq \{\}$ then $member\mathcal{O}(\mathcal{O}) \in \mathcal{O}$;
- if $\mathcal{F} \neq \{\}$ then $member\mathcal{F}(\mathcal{F}) \in \mathcal{F}$;
- if $turn(i) = S$ then $S \neq \{\}$.

Basically, canonicity imposes the minimal requirements that: *updt* can only update by enlarging; *sel*, $member\mathcal{O}$ and $member\mathcal{F}$ can only return some element in the input set, if this is non-empty; *turn* picks a set only if this is non-empty.

Example 4.1. The following choices of parameters are canonical:

1. for $R, C, D \subseteq \mathcal{L}$, $\sigma \in \mathcal{L}$: $f_{DbyC}(R, C) = (R \cap C = \{\})$
   $f_{DbyD}(R, D) = R$
   $f_{CbyD}(\sigma, D) = (\sigma \notin D)$
   $f_{CbyC}(R, C) = false$;
2. $updt(\mathcal{F}, S) = \mathcal{F}$;
3. for $S \subseteq \mathcal{L}$: if $S = \{\}$ then $sel(S) = \sigma \in S$ (namely *sel* returns any element in the input set)

$$turn(i) = \begin{cases} \mathcal{P}_i & \text{if } \mathcal{P}_i \neq \{\} \\ \mathcal{O}_i & \text{if } \mathcal{P}_i = \{\} \text{ and } \mathcal{O}_i \neq \{\} \\ \mathcal{F}_i & \text{if } \mathcal{P}_i = \{\} \text{ and } \mathcal{O}_i = \{\} \text{ and } \mathcal{F}_i \neq \{\}; \end{cases}$$

for $SS \subseteq \wp(\mathcal{L})$: $member\mathcal{O}(SS) = S \in SS$ (namely $member\mathcal{O}$ returns any element in the input set)
$member\mathcal{F}(SS) = S \in SS$ (namely $member\mathcal{F}$ returns any element in the input set).

In the remainder we will assume that the update and implementation choice parameters are canonical.

Like IB-dispute derivations, the definition of X-dispute derivations relies upon a marking mechanism, according to the following notation.

Notation 1. (See [14].) Given $S \subseteq \mathcal{L}$

- S_u is the set of *unmarked* sentences in S;
- $m(\sigma, S)$ is the set S where $\sigma \in S$ *becomes marked*;
- $u(S)$ is S where the marked sentences are *unmarked*.

We will see that only sentences in $\mathcal{O}$ may be marked. Unless explicitly marked, sentences are unmarked.

The formal definition of *X-dispute derivations* is given below. An intuitive reading of the definition is given in Fig. 3.

[6] We will refer to these components as $\mathcal{P}_i, \mathcal{O}_i, \mathcal{F}_i$ respectively.

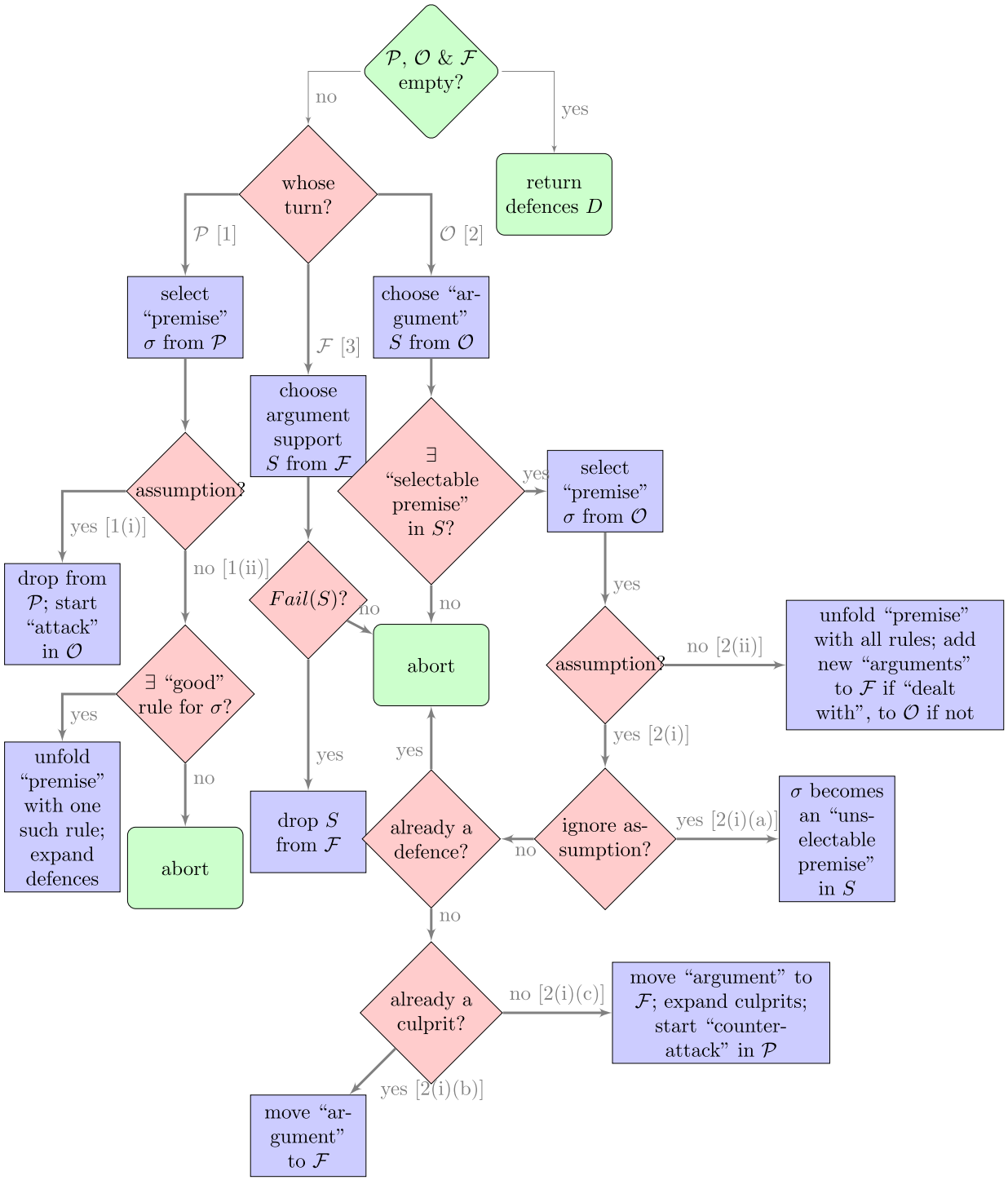


Fig. 3. A high-level, informal presentation of X-dispute-derivations (Definition 4.6) as a decision tree. Here, diamonds are decision points and boxes are commands. The numbers in square brackets correspond to cases in Definition 4.6. Green/rounded diamonds and boxes represent control information, implicit in Definition 4.6. Finally, there is an implicit arrow from each leaf box to the root diamond, to represent iteration.

Definition 4.6. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ be an ABA framework. A (successful) *X-dispute derivation of support* Δ *for sentence* $\delta \in \mathcal{L}$ w.r.t. parameters f_{DbyD}, f_{DbyC}, f_{CbyD}, f_{CbyC}, *updt*, *sel*, *member*$\mathcal{O}$, *member*$\mathcal{F}$, and *turn*, is a finite sequence of tuples

$$\langle \mathcal{P}_0, \mathcal{O}_0, D_0, C_0, \mathcal{F}_0 \rangle, \quad \ldots, \quad \langle \mathcal{P}_i, \mathcal{O}_i, D_i, C_i, \mathcal{F}_i \rangle, \quad \ldots, \quad \langle \mathcal{P}_n, \mathcal{O}_n, D_n, C_n, \mathcal{F}_n \rangle$$

where

$$\mathcal{P}_0 = \{\delta\} \quad D_0 = \mathcal{A} \cap \{\delta\}$$
$$\mathcal{O}_0 = C_0 = \mathcal{F}_0 = \{\}$$

$$\mathcal{P}_n = \mathcal{O}_n = \mathcal{F}_n = \{\}$$
$$\Delta = D_n$$

and for every $0 \leqslant i < n$:

1. If $turn(i) = \mathcal{P}_i$ and $sel(\mathcal{P}_i) = \sigma$ then
   (i) if $\sigma \in \mathcal{A}$, then

$$\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$$
$$\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\{\overline{\sigma}\}\}$$
$$D_{i+1} = D_i \quad C_{i+1} = C_i \quad \mathcal{F}_{i+1} = \mathcal{F}_i$$

   (ii) if $\sigma \notin \mathcal{A}$, then there exists $\sigma \leftarrow R \in \mathcal{R}$ such that $f_{DbyC}(R, C_i)$ and

$$\mathcal{P}_{i+1} = (\mathcal{P}_i - \{\sigma\}) \cup f_{DbyD}(R, D_i)$$
$$D_{i+1} = D_i \cup (\mathcal{A} \cap R)$$
$$C_{i+1} = C_i \quad \mathcal{O}_{i+1} = \mathcal{O}_i \quad \mathcal{F}_{i+1} = \mathcal{F}_i$$

2. If $turn(i) = \mathcal{O}_i$, $member\mathcal{O}(\mathcal{O}_i) = S$ and $sel(S_u) = \sigma$, then
   (i) if $\sigma \in \mathcal{A}$, then
      (a) either σ is ignored, i.e.

$$\mathcal{O}_{i+1} = (\mathcal{O}_i - \{S\}) \cup \{m(\sigma, S)\}$$
$$\mathcal{P}_{i+1} = \mathcal{P}_i \quad D_{i+1} = D_i \quad C_{i+1} = C_i \quad \mathcal{F}_{i+1} = \mathcal{F}_i$$

      (b) or $f_{CbyD}(\sigma, D_i)$ and $f_{CbyC}(\{\sigma\}, C_i)$ and

$$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$$
$$\mathcal{F}_{i+1} = updt(\mathcal{F}_i, \{u(S)\})$$
$$\mathcal{P}_{i+1} = \mathcal{P}_i \quad D_{i+1} = D_i \quad C_{i+1} = C_i$$

      (c) or $f_{CbyD}(\sigma, D_i)$ and not $f_{CbyC}(\{\sigma\}, C_i)$ and

$$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \quad C_{i+1} = C_i \cup \{\sigma\} \quad D_{i+1} = D_i \cup (\{\overline{\sigma}\} \cap \mathcal{A})$$
$$\mathcal{F}_{i+1} = updt(\mathcal{F}_i, \{u(S)\})$$
$$\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$$

   (ii) if $\sigma \notin \mathcal{A}$, then

$$\mathcal{O}_{i+1} = (\mathcal{O}_i - \{S\}) \cup \{(S - \{\sigma\}) \cup R | \sigma \leftarrow R \in \mathcal{R} \text{ and not } f_{CbyC}(R, C_i)\}$$
$$\mathcal{F}_{i+1} = updt(\mathcal{F}_i, \{(u(S) - \{\sigma\}) \cup R \mid \sigma \leftarrow R \in \mathcal{R} \text{ and } f_{CbyC}(R, C_i)\})$$
$$\mathcal{P}_{i+1} = \mathcal{P}_i \quad D_{i+1} = D_i \quad C_{i+1} = C_i$$

3. If $turn(i) = \mathcal{F}_i$ and $member\mathcal{F}(\mathcal{F}_i) = S$ and $Fail(S)$ then

$$\mathcal{F}_{i+1} = \mathcal{F}_i - \{S\}$$
$$\mathcal{O}_{i+1} = \mathcal{O}_i \quad \mathcal{P}_{i+1} = \mathcal{P}_i \quad D_{i+1} = D_i \quad C_{i+1} = C_i$$

Intuitively, X-dispute derivations start by initialising the (data structures for the) players ($\mathcal{P}$, $\mathcal{O}$, $\mathcal{F}$), defences (D) and culprits (C): the proponent $\mathcal{P}$ is set the task of "proving" and "defending" the sentence δ, this sentence immediately becomes a defence if it is an assumption, and everything else is empty. Then, X-dispute derivations proceed in steps, until (see also Fig. 3) there is nothing left to dispute ($\mathcal{P}$, $\mathcal{O}$ and $\mathcal{F}$ are all empty), in which case the accumulated set of defences is returned as output in support of the input sentence. At each (non-final) step, a player is chosen (via *turn*). If this is $\mathcal{F}$ (case 3), then one of its elements is chosen (via $member\mathcal{F}$). This represent an argument constructed by the opponent, for which *Fail*

must hold: if it does, then this argument is eliminated from $\mathcal{F}$, otherwise no case in the definition of X-dispute derivations applies, and thus the derivation cannot be continued (represented as 'abort' in Fig. 3). If the chosen player is $\mathcal{P}$ (case 1), then one of its elements is selected (via *sel*): this is basically a premise in one of the arguments being constructed by the proponent. If this premise is an assumption (case 1(i)), this is dropped and the opponent starts attacking it, by starting constructing arguments for its contrary. Necessarily, this premise is already included amongst the defences (by how these are initialised and then expanded in case 1(ii)). If the selected premise is not an assumption, then it needs to be expanded into a "more complete" argument, using a "good" rule, namely a rule not containing any culprits in its body (this is checked using f_{DbyC}): if none exists, then the derivation cannot be continued, else the premise is unfolded using one such chosen rule (but disregarding assumptions in its body already in D – as determined by f_{DbyD}) and the set of defences is enlarged with the assumptions in the body of this rule. Finally, if the chosen player is $\mathcal{O}$ (case 2), an opponent argument under construction is chosen (via *member*$\mathcal{O}$), and a premise is selected in its unmarked part (via *sel*): if no unmarked premise is left, and so none can be selected, the derivation cannot be continued, otherwise there are two subcases: the premise is an assumption (case 2(i)) or not (case 2(ii)). In the second subcase, the selected premise needs to be unfolded in all possible ways, to generate all possible "more complete" argument: those new arguments having some existing culprits in their body (as dictated by f_{CbyC}) can be safely ignored as already "dealt with", and moved onto $\mathcal{F}$ to be checked at a later stage, the others need to be further pursued (in $\mathcal{O}$). In the first subcase (2(i)), there are three possibilities: the assumption premise can be ignored (case 2(i)(a)) and become marked (this means that it won't be chosen as a culprit), or it can be chosen as a culprit (but then it cannot already be a defence, as determined by f_{CbyD}). This culprit can be an existing one (as determined by f_{CbyC}, case 2(i)(b)), in which case the argument under consideration can be deemed to be "dealt with" and passed on to $\mathcal{F}$, or a brand-new culprit (as determined by f_{CbyC}, case (2(i)(c)), in which case it needs to be passed on to $\mathcal{F}$, added to the culprits, as well as defended against by the proponent, which starts a new argument against this assumption (and for its contrary).

Example 4.2. Consider the ABA framework in Example 3.1 and the (canonical) choices of parameters in Example 4.1. Then, the GB-dispute derivation in Example 3.1 corresponds to a X-dispute derivation of support $\{a, c\}$ for p. We copy this dispute derivation below adding the $\mathcal{F}$ component (always empty) and noting the appropriate case applied to obtain each (non-initial) step:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	Note
0	$\{p\}$	{}	{}	{}	{}	
1	$\{a\}$	{}	$\{a\}$	{}	{}	by 1(ii)
2	{}	$\{\{q\}\}$	$\{a\}$	{}	{}	by 1(i)
3	{}	$\{\{b\}\}$	$\{a\}$	{}	{}	by 2(ii)
4	$\{r\}$	{}	$\{a\}$	$\{b\}$	{}	by 2(i)(c)
5	$\{c\}$	{}	$\{a, c\}$	$\{b\}$	{}	by 1(ii)
6	{}	$\{\{s\}\}$	$\{a, c\}$	$\{b\}$	{}	by 1(i)
7	{}	{}	$\{a, c\}$	$\{b\}$	{}	by 2(ii)

Note that no marking is performed in this derivation since case 2(i)(a) is never applied. As a consequence, no unmarking is ever applied either.

Note that *turn* does not perform a selection of player. Indeed, for example, case 2(i)(c) amounts to the choice of a culprit in, and thus a counter-attack against, a (possibly incomplete) argument of the opponent, and will typically be played by the proponent. Also, case 3 may be played by the proponent (if it is trying to discredit some of the attacks by the opponent) or by the opponent (if this is checking its own arguments).

Note that cases 1(ii) and 2(i)(a)/2(i)(c) rely upon (non-deterministic) choices of a rule (case 1(ii)) and whether to ignore an assumption (case 2(i)(a)) or not (case 2(i)(c)). These choices provide backtracking points in the implementation of dispute derivations, in that alternatives for these choices may need to be explored when trying to build a X-dispute-derivation.

Example 4.3. Consider the choices of parameters in Example 4.2 and the ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ with

- $\mathcal{R} = \{p \leftarrow q;\ p \leftarrow a;\ r \leftarrow b, c;\ t \leftarrow\}$;
- $\mathcal{A} = \{a, b, c\}$;
- $\bar{a} = r$, $\bar{b} = s$, $\bar{c} = t$.

The following is a failed attempt at finding a X-dispute derivation for p:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	Note
0	$\{p\}$	{}	{}	{}	{}	
1	$\{q\}$	{}	{}	{}	{}	by 1(ii)

Indeed, no case in the definition of X-dispute derivation can be applied in step 2, and $\mathcal{P}_1 \neq \{\}$. The failure here is due to the choice of $p \leftarrow q$ in step 1. The following is another failed attempt at finding a X-dispute derivation for p, after backtracking on the rule choice:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	Note
0	$\{p\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	
$1'$	$\{a\}$	$\{\}$	$\{a\}$	$\{\}$	$\{\}$	by 1(ii)
2	$\{\}$	$\{\{r\}\}$	$\{a\}$	$\{\}$	$\{\}$	by 1(i)
3	$\{\}$	$\{\{b, c\}\}$	$\{a\}$	$\{\}$	$\{\}$	by 2(ii)
4	$\{s\}$	$\{\}$	$\{a\}$	$\{b\}$	$\{\}$	by 2(i)(c) – with $sel(\{b, c\}) = b$

Indeed, again no case in the definition of X-dispute derivation can be applied in step 4. The failure here is due to the choice of b as a culprit in step 4. The following continuation, from step 3, of the earlier sequence is a (successful) X-dispute derivation for p, after backtracking on the choice of not ignoring b:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	Note
$4'$	$\{\}$	$\{\{c\}\}$	$\{a\}$	$\{\}$	$\{\}$	by 2(i)(a) – with $sel(\{b, c\}) = b$
5	$\{t\}$	$\{\}$	$\{a\}$	$\{c\}$	$\{\}$	by 2(i)(c)
6	$\{\}$	$\{\}$	$\{a\}$	$\{c\}$	$\{\}$	by 1(ii)

5. Soundness results for X-dispute derivations

We will first consider results for generic ABA frameworks, but w.r.t. specific choices of some of the parameters, and then results for the specific instance of ABA for logic programming [3].

5.1. Generic ABA frameworks

Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ be a (flat) ABA framework.

Definition 5.1. The following will be referred to as *GB-choices of parameters*:

1. $f_{DbyC}(R, C) = (R \cap C = \{\})$
   $f_{DbyD}(R, D) = R$
   $f_{CbyD}(\sigma, D) = (\sigma \notin D)$
   $f_{CbyC}(R, C) = false$
2. $updt(\mathcal{F}, S) = \mathcal{F}$
3. any canonical choice for the implementation choice parameters.

Trivially, GB-choices of parameters are canonical choices. Note that Example 4.2 used these GB-choices of parameters. X-dispute derivations for these choices of parameters correspond to the GB-dispute derivations of [14]:

Proposition 5.1 *(X-dispute derivations vs GB-dispute derivations). Let $\Delta \subseteq \mathcal{A}$ and $\delta \in \mathcal{L}$. There is a X-dispute derivation of support Δ for δ w.r.t. the GB-choices of parameters iff there is a GB-dispute derivation of defence set Δ for δ.*

This result is an immediate consequence of the definitions (of X- and GB-dispute derivations, see Definition A.1 in Appendix A for a recap of the latter) and can be easily seen by instantiating X-dispute derivations for the GB-choices of parameters. The original GB- and the X-dispute derivations for the GB-choices of parameters are identical, except for the $\mathcal{F}$ component and the use of marking in X-dispute derivations, both absent in GB-dispute derivations but playing no role in (this instance of) X-dispute derivations due to the notion of *updt* in the GB-choices of parameters.

Definition 5.2. The following will be referred to as *AB-choices of parameters*:

1. $f_{DbyC}(R, C) = (R \cap C = \{\})$
   $f_{DbyD}(R, D) = R - D$
   $f_{CbyD}(\sigma, D) = (\sigma \notin D)$
   $f_{CbyC}(R, C) = (R \cap C \neq \{\})$
2. $updt(\mathcal{F}, S) = \mathcal{F}$
3. any canonical choice for the implementation choice parameters.

Trivially, AB-choices of parameters are canonical choices. X-dispute derivations for AB-choices of parameters correspond to the AB-dispute derivations of [14]:

Proposition 5.2 *(X-dispute derivations vs AB-dispute derivations). Let $\Delta \subseteq \mathcal{A}$ and $\delta \in \mathcal{L}$. There is a X-dispute derivation of support Δ for δ w.r.t. the AB-choices of parameters iff there is an AB-dispute derivation of defence set Δ for δ.*

This result is an immediate consequence of the definitions (of X- and AB-dispute derivations, see Definition A.2 in Appendix A for a recap of the latter) and can be easily seen by instantiating X-dispute derivations for the AB-choices of parameters. Again, the original notion of AB-dispute derivation ignores $\mathcal{F}$ and marking.

Note that AB-choices differ from GB-choices only as far as f_{DbyD} and f_{CbyC} are concerned. Examples of the effects of these new definitions upon X-dispute derivations can be seen in Example 3.2: if extended with an empty $\mathcal{F}$ component at all steps, the derivations given therein are X-dispute derivations w.r.t. AB-choices of parameters.

Definition 5.3. The following will be referred to as *IB-choices of parameters*:

1. $f_{DbyC}(R, C) = (R \cap C = \{\})$
   $f_{DbyD}(R, D) = R - D$
   $f_{CbyD}(\sigma, D) = (\sigma \notin D)$
   $f_{CbyC}(R, C) = (R \cap C \neq \{\})$
2. $updt(\mathcal{F}, S) = \mathcal{F} \cup S$
3. any canonical choice for the implementation choice parameters.

Trivially, IB-choices of parameters are canonical choices. X-dispute derivations for these choices of parameters are identical to the IB-dispute derivations of [14]:

Proposition 5.3 *(X-dispute derivations vs IB-dispute derivations). Let $\Delta \subseteq \mathcal{A}$ and $\delta \in \mathcal{L}$. Any X-dispute derivation of support Δ for δ w.r.t. the IB-choices of parameters is an IB-dispute derivations of ideal support Δ for δ and vice versa.*

Again, this result can be proven directly by using the definitions (and using a variant of IB-dispute derivations given in Appendix A). Since IB-dispute derivations deploy $\mathcal{F}$ and marking, we obtain a direct correspondence in this case.

Note that IB-choices differ from AB-choices only as far as *updt* is concerned. An example of the effects of this new definition upon X-dispute derivations can be seen in Example 3.3: the derivation given therein is a X-dispute derivation w.r.t. IB-choices of parameters.

Note that, for both GB- and AB-choices of parameters, case 3 in Definition 4 never applies, as $\mathcal{F}$ will always be empty by definition of *updt* for these choices of parameters, since initially $\mathcal{F}$ is empty in X-dispute derivations.

Note that f_{CbyD} and f_{DbyC} are defined in the same way for GB-, AB- and IB-choices of parameters. Basically, these two forms of filtering ensure that the set of defence assumptions computed by dispute derivations is conflict-free and is thus an essential requirement for computing all semantics. We have chosen to represent these forms of filtering by means of parameters for uniformity, and to pave the way to modular experimentation with implementations. Moreover, note that f_{DbyD} and f_{CbyC} are defined in the same way for AB- and IB- choices, but differently for the GB-choice. In particular, the case of the GB-choices amounts to saying that GB-dispute derivations do not perform these kinds of filtering at all.

As a consequence of these correspondence results, all soundness and completeness results for GB-, AB- and IB-dispute derivations w.r.t. grounded, admissible and ideal semantics respectively [14] also hold for X-dispute derivations (for the appropriate choices of the parameters), namely:

Corollary 5.1 *(Soundness of X-dispute derivations w.r.t. grounded semantics). Given a X-dispute derivation of support $\Delta \subseteq \mathcal{A}$ for $\delta \in \mathcal{L}$ w.r.t. GB-choices of parameters,*

- *Δ is admissible and it is contained in the grounded set of assumptions*;
- *there exists $\Delta' \subseteq \Delta$ and an argument for δ supported by Δ'.*

This is a straightforward corollary of Proposition 5.1 above and of Theorem 4.2 in [14].

Corollary 5.2 *(Soundness of X-dispute derivations w.r.t. admissible semantics). Given a X-dispute derivation of support $\Delta \subseteq \mathcal{A}$ for $\delta \in \mathcal{L}$ w.r.t. AB-choices of parameters,*

- *Δ is admissible*;
- *there exists $\Delta' \subseteq \Delta$ and an argument for δ supported by Δ'.*

This is a straightforward corollary of Proposition 5.2 above and of Theorem 4.3 in [14].

Since every admissible set of assumption is contained in some *preferred* set of assumption (see Theorem 4.4 in [3]), the following is a direct consequence of Corollary 5.2:

Corollary 5.3 *(Soundness of X-dispute derivations w.r.t. preferred semantics). Given a X-dispute derivation of support $\Delta \subseteq \mathcal{A}$ for $\delta \in \mathcal{L}$ w.r.t. AB-choices of parameters, there exists a preferred set of assumptions Δ^* such that*

- $\Delta \subseteq \Delta^*$ *and* Δ^* *is preferred*;
- *there exists* $\Delta' \subseteq \Delta^*$ *and an argument for* δ *supported by* Δ'.

Corollary 5.4 *(Soundness of X-dispute derivations w.r.t. ideal semantics). Given a X-dispute derivation of support $\Delta \subseteq \mathcal{A}$ for $\delta \in \mathcal{L}$ w.r.t. IB-choices of parameters,*

- Δ *is contained in the ideal set of assumptions*;
- *there exists* $\Delta' \subseteq \Delta$ *and an argument for* δ *supported by* Δ'.

This is a straightforward corollary of Proposition 5.3 above and of Theorem 4.5 in [14].

5.2. Logic programming instance of ABA

Logic programming is an instance of ABA [3]. Indeed, every logic program P can be understood as a (flat) ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ where

- $\mathcal{L} = \{p, not\ p | p$ belongs to the Herbrand base of $P\}$,
- $\mathcal{R} = \{p \leftarrow B | p \leftarrow B$ is a ground instance of some $p' \leftarrow B' \in P\}$,
- $\mathcal{A} = \{not\ p | p$ belongs to the Herbrand base of $P\}$,
- $\overline{not\ p} = p$,

where *not* stands for negation as failure. Then, the admissible semantics in ABA amounts to the admissible semantics in logic programming [11,3], the grounded semantics in ABA amounts to the well-founded semantics in logic programming [30, 3], and the ideal semantics corresponds to the ideal semantics of [1]. Note that, as conventional when presenting the semantics of logic programming, we consider the grounding of the given logic program P. In the remainder of this section, for simplicity, we will assume that P, and any queries to be evaluated w.r.t. P, are ground. Also, unless otherwise stated, we will assume as given a logic programming instance of an ABA framework.

Definition 5.4. The following will be referred to as *LP-choices of parameters turn and sel*:

- $turn(\mathcal{P}, \mathcal{O}, \mathcal{F})$ is the non-empty element amongst $\mathcal{P}$ and $\mathcal{O}$ that has been most recently modified;
- $sel(S)$ is the most recently introduced element in S.

Example 5.1. Let $P = \{p \leftarrow not\ q, r; r \leftarrow not\ s; q \leftarrow not\ t; t \leftarrow not\ s\}$. Consider the fragment below of a X-dispute derivation (ignoring the marking for simplicity):

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$
0	$\{p\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{not\ q, r\}$	$\{\}$	$\{not\ q\}$	$\{\}$	$\{\}$
2	$\{r\}$	$\{\{q\}\}$	$\{not\ q\}$	$\{\}$	$\{\}$
3	$\{not\ s\}$	$\{\{q\}\}$	$\{not\ q, not\ s\}$	$\{\}$	$\{\}$
4	$\{\}$	$\{\{q\}, \{s\}\}$	$\{not\ q, not\ s\}$	$\{\}$	$\{\}$

This fragment does not use the LP-choice of parameter *turn*, since, at step 3, *turn* chooses $\mathcal{P}$, although $\mathcal{O}$ has been most recently modified. The following is a continuation of steps 0–2 above using the LP-choice of parameter *turn* (and GB-choices of parameters):

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$
$3'$	$\{r\}$	$\{\{not\ t\}\}$	$\{not\ q\}$	$\{\}$	$\{\}$
$4'$	$\{t, r\}$	$\{\}$	$\{not\ q\}$	$\{not\ t\}$	$\{\}$
5	$\{not\ s, r\}$	$\{\}$	$\{not\ q, not\ s\}$	$\{not\ t\}$	$\{\}$
6	$\{r\}$	$\{\{s\}\}$	$\{not\ q, not\ s\}$	$\{not\ t\}$	$\{\}$
7	$\{r\}$	$\{\}$	$\{not\ q, not\ s\}$	$\{not\ t\}$	$\{\}$
8	$\{not\ s\}$	$\{\}$	$\{not\ q, not\ s\}$	$\{not\ t\}$	$\{\}$
9	$\{\}$	$\{\{s\}\}$	$\{not\ q, not\ s\}$	$\{not\ t\}$	$\{\}$
10	$\{\}$	$\{\}$	$\{not\ q, not\ s\}$	$\{not\ t\}$	$\{\}$

Note that at step $4'$ a selection function *sel* choosing r in $\mathcal{P}$ would not be appropriate under the LP-choice of parameter *sel*, whereas *sel* choosing t, deployed therein, is.

To conclude the example, a corresponding X-dispute derivation using AB-choices of parameters and LP-choices of *turn* and *sel* is given by steps 0–2, 3′–4′, 5–7 extended with step 8′ below:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$
8′	{}	{}	{not q, not s}	{not t}	{}

The following result spells out the relationship between X-dispute derivations and SLDNF resolution in logic programming.

Proposition 5.4 *(X-dispute derivations vs SLDNF). Let $\Delta \subseteq \mathcal{A}$ and $\delta \in \mathcal{L}$. There is a X-dispute derivation (of support Δ) for δ w.r.t. GB-choices of parameters and LP-choices of parameters turn and sel iff there is a SLDNF derivation of δ w.r.t. selection function sel.*

The correspondence between the two kinds of derivations is illustrated by Example 5.1, where, for example, at step 2, the proof of *not q* is reduced to failure to prove q and, at step 4′, the disproof of *not t* is reduced to the proof of t. X-dispute derivations differ from the corresponding SLDNF derivations due to their use of marking, the accumulation of defences (D) and culprits (C), the use of the $\mathcal{F}$ component and the mixing of nested proofs, e.g. the mixing, at step 4′ of Example 5.1, of the proof of r to prove p and the proof of t to fail to prove q.

The following result spells out the relationship between X-dispute derivations and the abductive refutations of [23,11] (relying upon an abductive interpretation of negation as failure).

Proposition 5.5 *(X-dispute derivations vs abductive refutations). Let $\Delta \subseteq \mathcal{A}$ and $\delta \in \mathcal{L}$. There is a X-dispute derivation (of support Δ) for δ w.r.t. AB-choices of parameters and LP-choices of parameters turn and sel iff there is an abductive refutation from $(\delta, \{\})$ to $(\square, \Delta)$ w.r.t. sel.*[7]

X-dispute derivations differ from the corresponding abductive refutations due to their use of marking, the accumulation of culprits (C), the use of the $\mathcal{F}$ component and, as in the case of SLDNF, the mixing of nested proofs.

To the best of our knowledge, no computational mechanism exists for the ideal semantics in logic programming. IB-dispute derivations or X-dispute derivations, using IB-choices of parameters and, e.g., LP-choices of *turn* and *sel*, can be used for this purpose. Moreover, several other computational mechanisms can be obtained from X-dispute derivations (for GB-, AB- and IB-choices of parameters) to compute the well-founded, admissible and ideal semantics (respectively) in logic programming, for choices of *turn* and *sel* other than the LP-choices.

6. Structured X-dispute derivations

Structured X-dispute derivations are sequences of tuples of the form

$$\langle \mathcal{P}, \mathcal{O}, D, C, \mathcal{F}, Args, Att \rangle$$

where

- the elements D and C are defences and culprits, respectively, exactly as in X-dispute derivations (and the original AB-, GB- and IB-dispute derivations);
- $\mathcal{F}$ is as in X-dispute derivations;
- $\mathcal{P}$ and $\mathcal{O}$, as before, represent the state of the proponent and opponent, but they consist of "potential arguments" together with information about which "potential arguments" they attack;
- *Args* and *Att* hold, respectively, the currently computed ("potential") arguments and a binary relation between these arguments, corresponding to attacks currently identified.

Before we define these components formally, we define notions of potential argument and attack between potential arguments, adapted from [26].

Definition 6.1. A *potential argument* $A \vdash_S \sigma$ (*in favour of* $\sigma \in \mathcal{L}$ *supported by* A *given* S), is a proof for σ supported by $A \cup S$, as in Definition 2.1, with $A \subseteq \mathcal{A}$ and $S \subseteq \mathcal{L}$. A potential argument $A_1 \vdash_{S_1} \sigma_1$ *attacks* a potential argument $A_2 \vdash_{S_2} \sigma_2$ iff $\sigma_1 = \overline{\alpha}$ for some $\alpha \in A_2$.

Trivially, a potential argument $A \vdash_{\{\}} \sigma$ corresponds to an argument for σ supported by A as in conventional ABA (see Definition 2.1). Also, a potential argument $A \vdash_B \sigma$ with $B \neq \{\}$ but $B \subseteq \mathcal{A}$ corresponds to an argument for σ supported by $A \cup B$ in conventional ABA. We will refer to potential arguments corresponding to arguments in conventional ABA as *actual arguments*.

[7] Here, $\square$ stands for success as in standard logic programming.

Note that the same proof (tree) for a sentence may be represented by different potential arguments. For example, given an assumption $\alpha \in \mathcal{A}$, $\{\} \vdash_{\{\alpha\}} \alpha$ and $\{\alpha\} \vdash_{\{\}} \alpha$ are potential arguments for α (for the proof/tree with root and leaf α).

Note also that it may be possible to turn a potential argument $A \vdash_S \sigma$ into one, several or no actual arguments, depending on the rules in $\mathcal{R}$. For example, given $\mathcal{A} = \{a, b, c\}$ and $\mathcal{R} = \{p \leftarrow a, q\} \cup \mathcal{R}'$, depending on $\mathcal{R}'$, the potential argument $\{a\} \vdash_{\{q\}} p$ may be turned into

- no actual argument, if $\mathcal{R}' = \{\}$;
- one actual argument $\{a\} \vdash_{\{\}} p$, if $\mathcal{R}' = \{q \leftarrow\}$;
- two actual arguments $\{a, b\} \vdash_{\{\}} p$ and $\{a, c\} \vdash_{\{\}} p$, if $\mathcal{R}' = \{q \leftarrow b; q \leftarrow c\}$.

In the definition of the components of structured X-dispute derivations, as in [26], we adopt a labelling convention for potential arguments:

- *Args* consists of expressions of the form $l : A \vdash_S \sigma$ representing a potential argument $A \vdash_S \sigma$ labelled l;
- *Att* is a set of expressions of the form $l \rightsquigarrow l'$ indicating that the potential argument labelled l attacks the potential argument labelled l';
- $\mathcal{P}$ and $\mathcal{O}$ are sets of expressions of the form: $l : A \vdash_S \sigma \rightsquigarrow l'$ indicating a potential argument $A \vdash_S \sigma$ labelled l attacking another potential argument labelled l'.

For the purpose of labelling arguments, as for structured AB-dispute derivations [29], in the definition of structured X-dispute derivations we will use a procedure *newLabel*() that returns a fresh label every time it is invoked. Moreover, we will use a procedure *newlabel*(I) that returns a fresh label of the form $l(I)$ every time it is invoked with input I. We will use a special label $\emptyset$ in $l : A \vdash_S \sigma \rightsquigarrow \emptyset$ to indicate that the potential argument $A \vdash_S \sigma$, labelled l, is not attacking any known argument (but is instead introduced to support the initial claim σ).

The use of potential arguments explicitly in the $\mathcal{P}$ and $\mathcal{O}$ components renders the use of marking presented for X-dispute derivations unnecessary in *structured* X-dispute derivations. Indeed, given $A \vdash_S \sigma$, all sentences in A are marked and all sentences in S are unmarked, in the previous sense. For example, given $\mathcal{A} = \{a, b\}$ and rules $p \leftarrow a, q$ and $q \leftarrow b$ in $\mathcal{R}$, the potential argument $\{\} \vdash_{\{a,q\}} p$ in $\mathcal{O}$ in a structured X-dispute derivation corresponds to $\{a, q\}$ in $\mathcal{O}$ in a X-dispute derivation, with a and q both unmarked, whereas $\{a\} \vdash_{\{q\}} p$ corresponds to $\{m(a), q\}$, with a marked. Whereas marking in X-dispute derivations was introduced solely to encompass IB-dispute derivations, structured X-dispute derivations use the implicit marking afforded by the use of potential arguments to support the computation of actual arguments from potential arguments, e.g. by obtaining $\{a\} \vdash_{\{b\}} p$ and then $\{a, b\} \vdash_{\{\}} p$ in the earlier example.

Structured X-dispute derivations interleave the construction of arguments and their evaluation (w.r.t. a chosen semantics) and thus need to store potential arguments (in the components $\mathcal{P}$ and $\mathcal{O}$). Once these arguments are evaluated (w.r.t. the chosen semantics) they are eliminated from $\mathcal{P}$ or $\mathcal{O}$ and stored in *Args* (with *Att* also appropriately modified). For example, given $\mathcal{A} = \{a, b\}$ and $\mathcal{R} = \{p \leftarrow a, q; q \leftarrow b\}$, with $\bar{a} = r$ and $\bar{b} = s$, at some stage $\mathcal{P}$ and $\mathcal{O}$ may contain the potential arguments $\{a\} \vdash_{\{q\}} p$ and $\{\} \vdash_{\{r\}} r$ respectively, with the latter attacking the former even though neither is an actual argument. When the former is expanded to the actual argument $\{a, b\} \vdash_{\{\}} p$, this is removed from $\mathcal{P}$ and added to *Args*.

Structured X-dispute derivations are defined w.r.t. the same parameters as X-dispute derivations as well as a new parameter, *member*$\mathcal{P}$, to select (labelled) potential argument in $\mathcal{P}$. Moreover, in the case of structured X-dispute derivations, *member*$\mathcal{O}$ selects a labelled potential argument, rather than a set of sentences as in the case of X-dispute derivations.

Definition 6.2. Let Π be the set of all possible (labelled) potential arguments in $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$. Then

- *member*$\mathcal{O} : \wp(\Pi) \mapsto \Pi$;
- *member*$\mathcal{P} : \wp(\Pi) \mapsto \Pi$.

As for X-dispute derivations, choices of parameters can be canonical (see Definitions 4.2 and 4.5). For *member*$\mathcal{P}$ and (the new version of) *member*$\mathcal{O}$ canonicity amounts to requiring that

- if $\mathcal{O} \neq \{\}$ then *member*$\mathcal{O}(\mathcal{O}) \in \mathcal{O}$;
- if $\mathcal{P} \neq \{\}$ then *member*$\mathcal{P}(\mathcal{P}) \in \mathcal{P}$.

Definition 6.3. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ be an ABA framework. A (successful) *structured X-dispute derivation of support* Δ *and dialectical structure* (*Args*, *Att*) *for sentence* $\delta \in \mathcal{L}$ w.r.t. parameters f_{DbyD}, f_{DbyC}, f_{CbyD}, f_{CbyC}, *updt*, *sel*, *member*$\mathcal{P}$, *member*$\mathcal{O}$, *member*$\mathcal{F}$, and *turn*, is a finite sequence of tuples

$$\langle \mathcal{P}_0, \mathcal{O}_0, D_0, C_0, \mathcal{F}_0, Args_0, Att_0 \rangle, \ldots,$$
$$\langle \mathcal{P}_i, \mathcal{O}_i, D_i, C_i, \mathcal{F}_i, Args_i, Att_i \rangle, \ldots,$$

$\langle \mathcal{P}_n, \mathcal{O}_n, D_n, C_n, \mathcal{F}_n, Args_n, Att_n \rangle$

where

$$\mathcal{P}_0 = \{l_1 : \{\} \vdash_{\{\delta\}} \delta \rightsquigarrow \emptyset\} \quad \text{for } l_1 = newLabel() \quad D_0 = \mathcal{A} \cap \{\delta\}$$
$$\mathcal{O}_0 = C_0 = \mathcal{F}_0 = Args_0 = Att_0 = \{\}$$

$$\mathcal{P}_n = \mathcal{O}_n = \mathcal{F}_n = \{\}$$
$$\Delta = D_n \quad Args = Args_n \quad Att = Att_n$$

and for every $0 \leqslant i < n$:

1. If $turn(i) = \mathcal{P}_i$, $member\mathcal{P}(\mathcal{P}_i) = \pi$ where $\pi = (l : S_m \vdash_{S_u} \sigma_l \rightsquigarrow l')$ and $sel(S_u) = \sigma$ then
   (i) if $\sigma \in \mathcal{A}$, then

$$\mathcal{P}_{i+1} = (\mathcal{P}_i - \{\pi\}) \cup new\mathcal{P}$$
$$\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{l^* : \{\} \vdash_{\{\overline{\sigma}\}} \overline{\sigma} \rightsquigarrow l\}, \quad \text{for } l^* = newLabel()$$
$$Args_{i+1} = Args_i \cup newArgs$$
$$Att_{i+1} = Att_i \cup newAtt$$
$$D_{i+1} = D_i \quad C_{i+1} = C_i \quad \mathcal{F}_{i+1} = \mathcal{F}_i$$

   where $new\mathcal{P}$, $newArgs$ and $newAtt$ are as follows:

	$S_u - \{\sigma\} = \{\}$	$S_u - \{\sigma\} \neq \{\}$
$new\mathcal{P}$	$\{\}$	$\{l : (S_m \cup \{\sigma\}) \vdash_{(S_u - \{\sigma\})} \sigma_l \rightsquigarrow l'\}$
$newArgs$	$\{l : (S_m \cup \{\sigma\}) \vdash_{\{\}} \sigma_l\}$	$\{\}$
$newAtt$	$\{l \rightsquigarrow l'\}$	$\{\}$

   (ii) if $\sigma \notin \mathcal{A}$, then there exists $\sigma \leftarrow R \in \mathcal{R}$ such that $f_{DbyC}(R, C_i)$ and

$$\mathcal{P}_{i+1} = (\mathcal{P}_i - \{\pi\}) \cup new\mathcal{P}$$
$$D_{i+1} = D_i \cup (\mathcal{A} \cap R)$$
$$Args_{i+1} = Args_i \cup newArgs$$
$$newAtt_{i+1} = Att_i \cup newAtt$$
$$C_{i+1} = C_i \quad \mathcal{O}_{i+1} = \mathcal{O}_i \quad \mathcal{F}_{i+1} = \mathcal{F}_i$$

   where $new\mathcal{P}$, $newArgs$ and $newAtt$ are as follows:

	$(S_u - \{\sigma\}) \cup f_{DbyD}(R, D_i) = \{\}$	$(S_u - \{\sigma\}) \cup f_{DbyD}(R, D_i) \neq \{\}$
$new\mathcal{P}$	$\{\}$	$\{l : S'_m \vdash_{S'_u} \sigma_l \rightsquigarrow l'\}$
$newArgs$	$\{l : (S_m \cup R) \vdash_{\{\}} \sigma_l\}$	$\{\}$
$newAtt$	$\{l \rightsquigarrow l'\}$	$\{\}$ where $S'_m = S_m \cup (R - f_{DbyD}(R, D_i))$ and $S'_u = (S_u - \{\sigma\}) \cup f_{DbyD}(R, D_i)$

2. If $turn(i) = \mathcal{O}_i$, $member\mathcal{O}(\mathcal{O}_i) = \pi$ where $\pi = (l : S_m \vdash_{S_u} \sigma_l \rightsquigarrow l')$ and $sel(S_u) = \sigma$, then
   (i) if $\sigma \in \mathcal{A}$, then
      (a) either σ is ignored, i.e.

$$\mathcal{O}_{i+1} = (\mathcal{O}_i - \{\pi\}) \cup \{l : (S_m \cup \{\sigma\}) \vdash_{(S_u - \{\sigma\})} \sigma_l \rightsquigarrow l'\}$$
$$\mathcal{P}_{i+1} = \mathcal{P}_i \quad D_{i+1} = D_i \quad C_{i+1} = C_i \quad \mathcal{F}_{i+1} = \mathcal{F}_i$$
$$Args_{i+1} = Args_i \quad Att_{i+1} = Att_i$$

(b) or $f_{CbyD}(\sigma, D_i)$ and $f_{CbyC}(\{\sigma\}, C_i)$ and

$$\mathcal{O}_{i+1} = \mathcal{O}_i - \{\pi\}$$
$$\mathcal{F}_{i+1} = updt\big(\mathcal{F}_i, \{S_m \cup S_u\}\big)$$
$$Args_{i+1} = Args_i \cup \big\{l : \big(S_m \cup \{\sigma\}\big) \vdash_{(S_u - \{\sigma\})} \sigma_l\big\}$$
$$\mathcal{P}_{i+1} = \mathcal{P}_i \quad D_{i+1} = D_i \quad C_{i+1} = C_i$$

(c) or $f_{CbyD}(\sigma, D_i)$ and not $f_{CbyC}(\{\sigma\}, C_i)$ and

$$\mathcal{O}_{i+1} = \mathcal{O}_i - \{\pi\} \quad C_{i+1} = C_i \cup \{\sigma\} \quad D_{i+1} = D_i \cup (\{\overline{\sigma}\} \cap \mathcal{A})$$
$$\mathcal{F}_{i+1} = updt\big(\mathcal{F}_i, \{S_m \cup S_u\}\big)$$
$$\mathcal{P}_{i+1} = \mathcal{P}_i \cup \big\{l^* : \{\} \vdash_{\{\overline{\sigma}\}} \overline{\sigma} \rightsquigarrow l\big\}, \quad \text{for } l^* = newLabel()$$
$$Args_{i+1} = Args_i \cup \big\{l : \big(S_m \cup \{\sigma\}\big) \vdash_{(S_u - \{\sigma\})} \sigma_l\big\}$$
$$Att_{i+1} = Att_i \cup \{l \rightsquigarrow l'\}$$

(ii) if $\sigma \notin \mathcal{A}$, let
- $\mathcal{S}_f = \{R | \sigma \leftarrow R \in \mathcal{R}$ and $f_{CbyC}(R, C_i)\}$ and
- $\mathcal{S}_{nf} = \{R | \sigma \leftarrow R \in \mathcal{R}$ and not $f_{CbyC}(R, C_i)\}$

then, given that $l(R)$ is the outcome of $newLabel(R)$:

$$\mathcal{O}_{i+1} = \big(\mathcal{O}_i - \{\pi\}\big) \cup \big\{l(R) : S_m \vdash_{((S_u - \{\sigma\}) \cup R)} \sigma_l \rightsquigarrow l' | R \in \mathcal{S}_{nf}\big\}$$
$$\mathcal{F}_{i+1} = updt\big(\mathcal{F}_i, \big\{S_m \cup \big(\big(S_u - \{\sigma\}\big) \cup R\big) \mid R \in \mathcal{S}_f\big\}\big)$$
$$Args_{i+1} = Args_i \cup \big\{l(R) : S'_m \vdash_{S'_u} \sigma_l | R \in \mathcal{S}_f, \quad S'_m = S_m \cup (R \cap C_i), \quad S'_u = \big(S_u - \{\sigma\}\big) \cup (R - C_i)\big\}$$
$$Att_{i+1} = Att_i \cup \big\{l(R) \rightsquigarrow l' | R \in \mathcal{S}_f\big\}$$
$$\mathcal{P}_{i+1} = \mathcal{P}_i \quad D_{i+1} = D_i \quad C_{i+1} = C_i$$

3. If $turn(i) = \mathcal{F}_i$ and $member\mathcal{F}(\mathcal{F}_i) = S$ and $Fail(S)$ then

$$\mathcal{F}_{i+1} = \mathcal{F}_i - \{S\}$$
$$\mathcal{O}_{i+1} = \mathcal{O}_i \quad \mathcal{P}_{i+1} = \mathcal{P}_i \quad D_{i+1} = D_i \quad C_{i+1} = C_i$$
$$Args_{i+1} = Args_i \quad Att_{i+1} = Att_i$$

We will refer to $(Args, Att)$ and S as the *dialectical structure and support* (*respectively*) *computed by the structured X-dispute derivation*. We will refer to C_n as the *culprits computed by the structured X-dispute derivation*.

Structured X-dispute derivations can be given an analogous intuitive reading to X-dispute derivations, by extending the decision tree given in Fig. 3 as in Fig. 4. Algorithmically, the new decision tree accommodates the choice of a proponent argument in case 1, by $member\mathcal{P}$. Moreover, potential arguments (referred to as p-arguments in the figure) are explicitly manipulated in $\mathcal{P}$ and $\mathcal{O}$, so that unfolding the selected premise in (the unmarked support of) the chosen argument results, in case 1(ii), into creating a new argument $new\mathcal{P}$ and, in case 2(ii), into creating a bunch of new arguments (with labels $l(R)$). Note that, in case 1(ii), if this unfolding results into an argument with an empty (filtered) unmarked support then it is simply removed from $\mathcal{P}$ (and moved into *Args*) as, intuitively, this is an argument that is being successfully defended and can be put aside. Instead, in case 2(ii), any such argument, with an empty unmarked support, is kept in $\mathcal{O}$, and will cause for the derivation to be aborted at a later step, correctly (as it corresponds to an argument in whose support no culprit can be chosen). Note also that, in addition to the case where a new argument is started (to counter-attack an existing argument, in case 1(i)), similarly to X-dispute derivations, now $\mathcal{P}$ is also modified, in case 1(i), as a result of selecting an assumption premise and moving it into the marked part of the support. Again, as in case 1(ii), if as a result the unmarked part becomes empty, then the argument is moved from $\mathcal{P}$ into *Args*. Structured X-dispute derivations also operate upon the new component *Args* (not indicated in the figure for simplicity) as follows:

- in cases 1(i) and 1(ii), as a result of moving an argument with an empty unmarked support from $\mathcal{P}$;

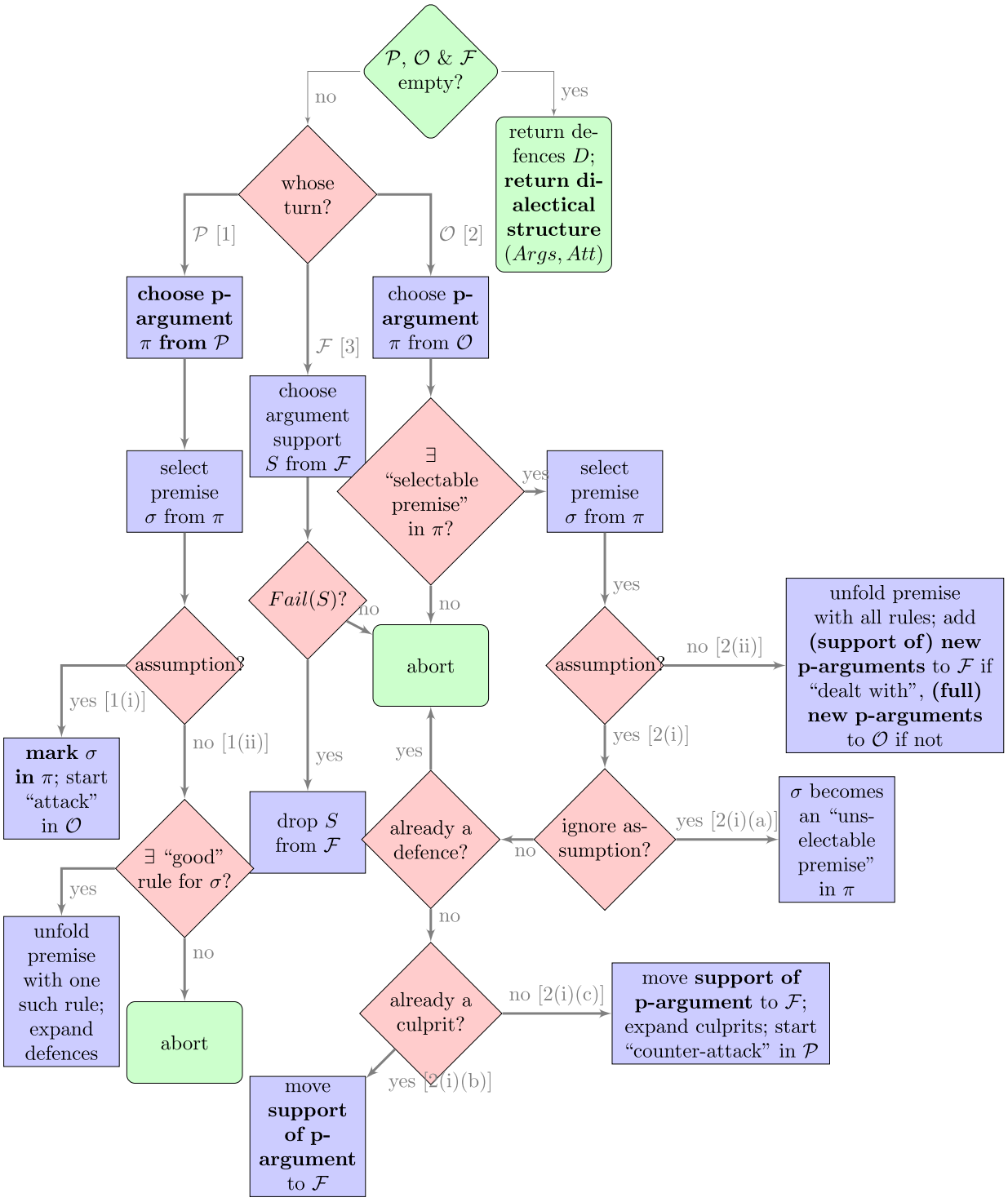


Fig. 4. A high-level, informal presentation of structured X-dispute-derivations (Definition 6.3) extending the presentation of X-dispute derivations in Fig. 3. The main extensions are indicated in boldface.

- in case 2(i)(b), as a result of filtering, identifying the selected premise as a culprit already being defeated, and thus allowing to disregard the argument as "dealt with";
- in case 2(i)(c), as a result of choosing a new culprit in the chosen argument and starting defeating it, and thus allowing to disregard the argument as "dealt with";
- in case 2(ii), as a result of identifying, in the newly unfolded arguments, already "filtered" arguments ($\mathcal{S}_f$).

$Step$	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	$Args$	Att
0	$\{l_1 : \{\} \vdash_{\{p\}} p \rightsquigarrow \emptyset\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{l_1 : \{\} \vdash_{\{a\}} p \rightsquigarrow \emptyset\}$	$\{\}$	$\{a\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
2	$\{\}$	$\{l_2 : \{\} \vdash_{\{q\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p\}$	$\{l_1 \rightsquigarrow \emptyset\}$
3	$\{\}$	$\{l'_2 : \{\} \vdash_{\{b\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p\}$	$\{l_1 \rightsquigarrow \emptyset\}$
4	$\{l_3 : \{\} \vdash_{\{r\}} r \rightsquigarrow l'_2\}$	$\{\}$	$\{a\}$	$\{b\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1\}$
5	$\{l_3 : \{\} \vdash_{\{c\}} r \rightsquigarrow l'_2\}$	$\{\}$	$\{a, c\}$	$\{b\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1\}$
6	$\{\}$	$\{l_4 : \{\} \vdash_{\{s\}} s\}$	$\{a, c\}$	$\{b\}$	$\{\}$	$Args_6$	Att_6
7	$\{\}$	$\{\}$	$\{a, c\}$	$\{b\}$	$\{\}$	$Args_6$	Att_6

Fig. 5. A structured X-dispute derivation for GB-choices of parameters for Example 6.1. Here l'_2 stands for the outcome of $newLabel(q \leftarrow b)$, and $l_1, \ldots, l_4$ are the outcome of (successive calls to) $newLabel()$. The non-initial steps are obtained as follows: step 1 by case 1(ii), step 2 by case 1(i), step 3 by case 2(ii), step 4 by case 2(i)(c), step 5 by case 1(ii), step 6 by case 1(i), step 7 by case 2(ii).

Every change to *Args* is naturally accompanied by a change to *Att*, to include all attacks from the newly added arguments to other arguments. Finally, note that the treatment of $\mathcal{F}$ is essentially unchanged, w.r.t. X-dispute derivations.

We conclude this section by illustrating the notion of structured X-dispute derivation with several examples. Here and in the remainder of the paper we will use the following terminology: given some choices of parameters C amongst GB-, AB- and IB-, for some specific canonical choices of *sel*, *member*$\mathcal{P}$, *member*$\mathcal{O}$, *member*$\mathcal{F}$ and *turn*, we say that some other choices of parameters (amongst GB-, AB- and IB-) *agree with* C if they adopt the same canonical choices of *sel*, *member*$\mathcal{P}$, *member*$\mathcal{O}$, *member*$\mathcal{F}$ and *turn* as C.

Example 6.1. Consider the ABA framework in Examples 3.1 and 4.2 and (GB-)choices of parameters as in Example 4.2 and in addition $memberP(SS) = S \in SS$. Then, Fig. 5 gives a structured X-dispute derivation of support $\{a, c\}$ and $(Args_6, Att_6)$ for p, where

- $Args_6 = \{l_1 : \{a\} \vdash_{\{\}} p,\ l'_2 : \{b\} \vdash_{\{\}} q,\ l_3 : \{c\} \vdash_{\{\}} r\}$
- $Att_6 = \{l_1 \rightsquigarrow \emptyset, l'_2 \rightsquigarrow l_1, l_3 \rightsquigarrow l'_2\}$

Exactly the same derivation is obtained for AB-choices of parameters (agreeing with the earlier choices in this example). Given the same ABA framework and (GB- or AB-)choices of parameters as above, the following is a failed attempt at finding a structured X-dispute derivation for q:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	*Args*	*Att*
0	$\{l_1 : \{\} \vdash_{\{q\}} q \rightsquigarrow \emptyset\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{l_1 : \{\} \vdash_{\{b\}} q \rightsquigarrow \emptyset\}$	$\{\}$	$\{b\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
2	$\{\}$	$\{l_2 : \{\} \vdash_{\{r\}} r \rightsquigarrow l_1\}$	$\{b\}$	$\{\}$	$\{\}$	$\{l_1 : \{b\} \vdash_{\{\}} q\}$	$\{l_1 \rightsquigarrow \emptyset\}$
3	$\{\}$	$\{l_2 : \{\} \vdash_{\{c\}} r \rightsquigarrow l_1\}$	$\{b\}$	$\{\}$	$\{\}$	$\{l_1 : \{b\} \vdash_{\{\}} q\}$	$\{l_1 \rightsquigarrow \emptyset\}$
4	$\{l_3 : \{\} \vdash_{\{s\}} s \rightsquigarrow l_2\}$	$\{\}$	$\{b\}$	$\{c\}$	$\{\}$	$\{l_1 : \{b\} \vdash_{\{\}} q,$ $l_2 : \{c\} \vdash_{\{\}} r\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l_2 \rightsquigarrow l_1\}$

This is not a successful structured X-dispute derivation as $\mathcal{P}_4$ is not empty. It is not possible to extend this sequence to a successful derivation.

Given the same ABA framework but IB-choices of parameters (agreeing with the earlier choices in this example), Fig. 6 shows a structured X-dispute derivation of support $\{a, c\}$ and $(Args_6, Att_6)$ for p. This is the same as the derivation in Fig. 5 except for the $\mathcal{F}$ component and the addition of a final step 8, obtained by applying case 3 in Definition 6.3. Note that $Fail(\{b\})$ holds because b cannot possibly belong to an admissible set of assumptions.

Example 6.2. Consider the ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ with

- $\mathcal{R} = \{p \leftarrow a, u;\ q \leftarrow b, r;\ q \leftarrow c, s;\ q \leftarrow c, t;\ u \leftarrow a;\ s \leftarrow;\ t \leftarrow d;\ t \leftarrow e\}$;
- $\mathcal{A} = \{a, b, c, d, e, f\}$;
- $\bar{a} = q$, $\bar{b} = f$, $\bar{c} = u$, $\bar{d} = v$, $\bar{e} = v$, $\bar{f} = v$.

Fig. 7 gives a structured X-dispute derivation of support $\{a, f\}$ and $(Args_{11}, Att_{11})$ for p, for AB-choices of parameters (and agreeing with the choices in Example 6.1). Note that (structured) X-dispute derivations manipulate multi-sets. For example, the support of the argument labelled l_1 in the figure has two occurrences of the sentence a. This is not a structured X-dispute derivation for GB-choices of parameters, as, for example, at step 6 filtering by defence a cannot be applied using f_{DbyD} as in GB-choices. Using IB-choices of parameters (agreeing with the other choices), a corresponding sequence can

$Step$	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	$Args$	Att
0	$\{l_1 : \{\} \vdash_{\{p\}} p \rightsquigarrow \emptyset\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{l_1 : \{\} \vdash_{\{a\}} p \rightsquigarrow \emptyset\}$	$\{\}$	$\{a\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
2	$\{\}$	$\{l_2 : \{\} \vdash_{\{q\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p\}$	$\{l_1 \rightsquigarrow \emptyset\}$
3	$\{\}$	$\{l'_2 : \{\} \vdash_{\{b\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p\}$	$\{l_1 \rightsquigarrow \emptyset\}$
4	$\{l_3 : \{\} \vdash_{\{r\}} r \rightsquigarrow l'_2\}$	$\{\}$	$\{a\}$	$\{b\}$	$\{\{b\}\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1\}$
5	$\{l_3 : \{\} \vdash_{\{c\}} r \rightsquigarrow l'_2\}$	$\{\}$	$\{a, c\}$	$\{b\}$	$\{\{b\}\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1\}$
6	$\{\}$	$\{l_4 : \{\} \vdash_{\{s\}} s\}$	$\{a, c\}$	$\{b\}$	$\{\{b\}\}$	$Args_6$	Att_6
7	$\{\}$	$\{\}$	$\{a, c\}$	$\{b\}$	$\{\{b\}\}$	$Args_6$	Att_6
8	$\{\}$	$\{\}$	$\{a, c\}$	$\{b\}$	$\{\}$	$Args_6$	Att_6

Fig. 6. A structured X-dispute derivation for IB-choices of parameters for Example 6.1. Step 8 is obtained by case 3.

$Step$	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	$Args$	Att
0	$\{l_1 : \{\} \vdash_{\{p\}} p \rightsquigarrow \emptyset\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{l_1 : \{\} \vdash_{\{a,u\}} p \rightsquigarrow \emptyset\}$	$\{\}$	$\{a\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
2	$\{l_1 : \{a\} \vdash_{\{u\}} p \rightsquigarrow \emptyset\}$	$\{l_2 : \{\} \vdash_{\{q\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
3	$\{l_1 : \{a\} \vdash_{\{u\}} p \rightsquigarrow \emptyset\}$	$\{l'_2 : \{\} \vdash_{\{b,r\}} q \rightsquigarrow l_1,$ $l''_2 : \{\} \vdash_{\{c,s\}} q \rightsquigarrow l_1,$ $l'''_2 : \{\} \vdash_{\{c,t\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
4	$\{l_1 : \{a\} \vdash_{\{u\}} p \rightsquigarrow \emptyset\}$	$\{l'_2 : \{\} \vdash_{\{b,r\}} q \rightsquigarrow l_1,$ $l^{**}_2 : \{\} \vdash_{\{c\}} q \rightsquigarrow l_1,$ $l'''_2 : \{\} \vdash_{\{c,t\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
5	$\{l_1 : \{a\} \vdash_{\{u\}} p \rightsquigarrow \emptyset,$ $l_3 : \{\} \vdash_{\{u\}} u \rightsquigarrow l^{**}_2\}$	$\{l'_2 : \{\} \vdash_{\{b,r\}} q \rightsquigarrow l_1,$ $l'''_2 : \{\} \vdash_{\{c,t\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{c\}$	$\{\}$	$\{l^{**}_2 : \{c\} \vdash_{\{\}} q\}$	$\{l^{**}_2 \rightsquigarrow l_1\}$
6	$\{l_3 : \{\} \vdash_{\{u\}} u \rightsquigarrow l^{**}_2\}$	$\{l'_2 : \{\} \vdash_{\{b,r\}} q \rightsquigarrow l_1,$ $l'''_2 : \{\} \vdash_{\{c,t\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{c\}$	$\{\}$	$\{l^{**}_2 : \{c\} \vdash_{\{\}} q,$ $l_1 : \{a, a\} \vdash_{\{\}} p\}$	$\{l^{**}_2 \rightsquigarrow l_1,$ $l_1 \rightsquigarrow \emptyset\}$
7	$\{l_3 : \{\} \vdash_{\{u\}} u \rightsquigarrow l^{**}_2,$ $l_5 : \{\} \vdash_{\{f\}} f \rightsquigarrow l'_2\}$	$\{l'''_2 : \{\} \vdash_{\{c,t\}} q \rightsquigarrow l_1\}$	$\{a, f\}$	$\{c, b\}$	$\{\}$	$\{l^{**}_2 : \{c\} \vdash_{\{\}} q,$ $l_1 : \{a, a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{r\}} q\}$	$\{l^{**}_2 \rightsquigarrow l_1,$ $l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1\}$
8	$\{l_3 : \{\} \vdash_{\{u\}} u \rightsquigarrow l^{**}_2\}$	$\{l'''_2 : \{\} \vdash_{\{c,t\}} q \rightsquigarrow l_1,$ $l_6 : \{\} \vdash_{\{v\}} v \rightsquigarrow l_5\}$	$\{a, f\}$	$\{c, b\}$	$\{\}$	$\{l^{**}_2 : \{c\} \vdash_{\{\}} q,$ $l_1 : \{a, a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{r\}} q,$ $l_5 : \{f\} \vdash_{\{\}} f\}$	$\{l^{**}_2 \rightsquigarrow l_1,$ $l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1,$ $l_5 \rightsquigarrow l'_2\}$
9	$\{l_3 : \{\} \vdash_{\{u\}} u \rightsquigarrow l^{**}_2\}$	$\{l'''_2 : \{\} \vdash_{\{c,t\}} q \rightsquigarrow l_1\}$	$\{a, f\}$	$\{c, b\}$	$\{\}$	$\{l^{**}_2 : \{c\} \vdash_{\{\}} q,$ $l_1 : \{a, a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{r\}} q,$ $l_5 : \{f\} \vdash_{\{\}} f\}$	$\{l^{**}_2 \rightsquigarrow l_1,$ $l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1,$ $l_5 \rightsquigarrow l'_2\}$
10	$\{l_3 : \{\} \vdash_{\{u\}} u \rightsquigarrow l^{**}_2\}$	$\{\}$	$\{a, f\}$	$\{c, b\}$	$\{\}$	$\{l^{**}_2 : \{c\} \vdash_{\{\}} q,$ $l_1 : \{a, a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{r\}} q,$ $l_5 : \{f\} \vdash_{\{\}} f,$ $l'''_2 : \{c\} \vdash_{\{t\}} q\}$	$\{l^{**}_2 \rightsquigarrow l_1,$ $l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1,$ $l_5 \rightsquigarrow l'_2,$ $l'''_2 \rightsquigarrow l_1\}$
11	$\{\}$	$\{\}$	$\{a, f\}$	$\{c, b\}$	$\{\}$	$\{l^{**}_2 : \{c\} \vdash_{\{\}} q,$ $l_1 : \{a, a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{r\}} q,$ $l_5 : \{f\} \vdash_{\{\}} f,$ $l'''_2 : \{c\} \vdash_{\{t\}} q,$ $l_3 : \{a\} \vdash_{\{\}} u\}$	$\{l^{**}_2 \rightsquigarrow l_1,$ $l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1,$ $l_5 \rightsquigarrow l'_2,$ $l'''_2 \rightsquigarrow l_1,$ $l_3 \rightsquigarrow l^{**}_2\}$

Fig. 7. A structured X-dispute derivation for Example 6.2. Here, the non-initial steps are obtained as follows: step 1 by case 1(ii), step 2 by case 1(i), step 3 by case 2(ii), step 4 by case 2(ii) (on the argument labelled by l'''_2) step 5 by case 2(i)(c), step 6 by case 1(ii) (filtering by defence a in the argument labelled by l_1), step 7 by case 2(i)(c) (choosing culprit b in the argument labelled by l'_2), step 8 by case 1(i) (on the argument labelled by l_5), step 9 by case 2(ii) (on the argument labelled by l_6), step 10 by case 2(i)(b) (filtering by culprit c in the argument labelled by l'''_2), step 11 by case 1(i) (filtering by defence a in the argument labelled by l_3).

be obtained, but with different $\mathcal{F}$ components, with, in particular, $\mathcal{F}_{11} = \{\{c\}, \{b, r\}, \{c, t\}\}$. Since $Fail(\{c\})$ does not hold (as $\{c, e\}$ is admissible), this sequence cannot be extended to a successful structured X-dispute derivation for IB-choices of parameters.

7. Soundness results for structured X-dispute derivations

We will first (Section 7.1) consider soundness in the same sense as the original AB-, GB- and IB-dispute derivations and X-dispute derivations, namely we will prove that the support computed by structured X-dispute derivations is acceptable (i.e. admissible, grounded, ideal) w.r.t. appropriate choices of the parameters. We will obtain this result as a corollary of a one-to-one correspondence between structured X-dispute derivations and X-dispute derivations. Then (Section 7.2), we will study the soundness of structured X-dispute derivations as far as the computed dialectical structure $(Args, Att)$ is concerned.

7.1. Soundness of support

There is a one-to-one correspondence between X-dispute derivations and structured X-dispute derivations.

Theorem 7.1 *(Structured X-dispute derivations vs X-dispute derivations). Let $\Delta \subseteq \mathcal{A}$ and $\delta \in \mathcal{L}$. There exists a structured X-dispute derivation of support Δ and $(Args, Att)$ for δ, for some $(Args, Att)$ and w.r.t. some choices of parameters, iff there exists a X-dispute derivation of support Δ for δ, w.r.t. some choices of parameters.*

The proof of this theorem is in Appendix B.1.

The following result sanctions the soundness of structured X-dispute derivations as far as the computed set of assumptions is concerned. It is a straightforward consequence of Theorem 7.1 and Corollaries 5.1, 5.2, 5.3 and 5.4 in Section 5.

Corollary 7.1 *(Soundness of structured X-dispute derivations – support). If there exists a structured X-dispute derivation of support Δ and $(Args, Att)$ for δ (for some $(Args, Att)$)*

- *w.r.t. GB-choices of parameters then*
  - *Δ is admissible and it is contained in the grounded set of assumptions, and*
  - *there exists $\Delta' \subseteq \Delta$ and an argument for δ supported by Δ';*
- *w.r.t. AB-choices of parameters then*
  - *Δ is admissible,*
  - *there exists $\Delta^* \supseteq \Delta$ such that Δ^* is preferred, and*
  - *there exists $\Delta' \subseteq \Delta$ and an argument for δ supported by Δ';*
- *w.r.t. IB-choices of parameters then*
  - *Δ is contained in the ideal set of assumptions, and*
  - *there exists $\Delta' \subseteq \Delta$ and an argument for δ supported by Δ'.*

7.2. Soundness of dialectical structure

We define a mapping between the dialectical structure $(Args, Att)$ computed by structured X-dispute derivations and, through several steps, trees that, for appropriate choices of parameters, are *grounded/admissible/ideal dispute trees* (see [13, 14] and, for an overview of these trees, Section 2).

First note that $(Args, Att)$ corresponds to a tree, as follows:

Definition 7.1. Let $(Args, Att)$ be the dialectical structure computed by a structured X-dispute derivation for some sentence. $\mathcal{T}^*(Args, Att)$ is the tree with (labelled potential) arguments in $Args$ as nodes such that

- the root of $\mathcal{T}^*(Args, Att)$ is the potential argument in $Args$ with label l such that $l \leadsto \emptyset \in Att$ (trivially, there is exactly one such l for $(Args, Att)$ computed by a structured X-dispute derivation), and
- if a node in $\mathcal{T}^*(Args, Att)$ is an argument in $Args$ with label l_N, then the node has as children all the arguments in $Args$ with label l_M such that $l_M \leadsto l_N \in Att$.

$\mathcal{T}^*(Args, Att)$ for the dialectical structures computed in Examples 6.1 and 6.2 are given in Fig. 8 (left) and Fig. 8 (right) respectively.

Trees $\mathcal{T}^*(Args, Att)$ are in general not dispute trees, in the first place because of the presence in them of non-actual arguments, (e.g. the arguments labelled l'_2 and l'''_2 in Fig. 8 (right)). We derive trees with actual arguments only from a dialectical structure $(Args^a, Att^a)$ (that we call *actual dialectical structure*, see Section 7.2.1) obtained by expanding potential arguments in $(Args, Att)$ into actual arguments, if any can be obtained from them. Then, we map this actual dialectical structure into a *pruned dialectical forest* of trees (Section 7.2.2), to prove our soundness result for the grounded semantics.

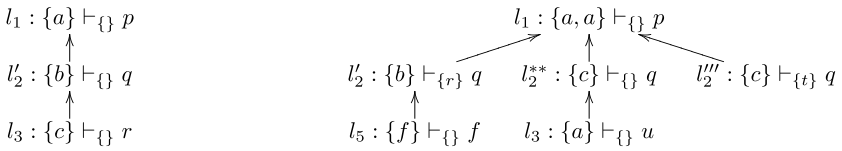


Fig. 8. Trees $\mathcal{T}^*(Args, Att)$ for the dialectical structures $(Args, Att)$ computed in Examples 6.1 (left) and 6.2 (right).

The pruning amounts to removing argument labels. The forest includes a tree for (an argument for) the input sentence for the given structured X-dispute derivation, as well as trees for (arguments for) other sentences, possibly introduced to defend against non-actual attacks computed in the derivation. Finally, we define the notion of *expanded dialectical forest* (Section 7.2.3), to prove our soundness result for the admissible and ideal semantics. The expansion amounts to "undoing" the effects of filtering during the derivation, by "hanging" sub-trees below arguments dealt with by filtering.

7.2.1. Actual dialectical structure

This is obtained as an expansion of $(Args, Att)$, defined as follows:

Definition 7.2. Given a potential argument $A \vdash_S \sigma$ with $S \neq \{\}$, a *proof for σ supported by $A \cup B$ and expanding $A \vdash_S \sigma$* is a proof for σ supported by $A \cup B$ such that $B = \bigcup_{\sigma' \in S} \zeta(\sigma')$, where, for a given σ', $\zeta(\sigma')$ is a set of assumptions such that there is a proof for σ' supported by $\zeta(\sigma')$.

In Example 6.2, there are two proofs for q expanding $\{c\} \vdash_{\{t\}} q$, supported by $\{c, d\}$ and $\{c, e\}$ respectively. However, there is no proof for q expanding $\{b\} \vdash_{\{r\}} q$. Note that there is only one possible proof for σ supported by $A \cup B$ and expanding $A \vdash_S \sigma$ if $S \subseteq \mathcal{A}$.

Definition 7.3. Let $(Args, Att)$ be the dialectical structure computed by a structured X-dispute derivation for some sentence. The *actual dialectical structure* $Actual(Args, Att)$ is $(Args^a, Att^a)$ such that

- $Args^a = \{l : A \vdash_{\{\}} \sigma \,|\, l : A \vdash_{\{\}} \sigma \in Args\} \cup$
  $\{l_{\langle S,S' \rangle} : A \cup S' \vdash_{\{\}} \sigma \,|\, l : A \vdash_S \sigma \in Args, S \neq \{\}$ and
  there is a proof for σ supported by $A \cup S'$ and expanding $A \vdash_S \sigma\}$;
- $Att^a = \{l \rightsquigarrow \emptyset, l \rightsquigarrow l' \in Att \,|\, l : A \vdash_{\{\}} \sigma, l' : A' \vdash_{\{\}} \sigma' \in Args^a \cap Args\} \cup$
  $\{l \rightsquigarrow l'_{\langle S,S' \rangle} \,|\, l : A \vdash_{\{\}} \sigma \in Args^a \cap Args, l'_{\langle S,S' \rangle} \in Args^a \setminus Args$ and $l \rightsquigarrow l' \in Att\} \cup$
  $\{l'_{\langle S,S' \rangle} \rightsquigarrow l \,|\, l : A \vdash_{\{\}} \sigma \in Args^a \cap Args, l'_{\langle S,S' \rangle} \in Args^a \setminus Args$ and $l' \rightsquigarrow l \in Att\}$.

Intuitively, the construction of $Actual(Args, Att)$ expands the support of potential, non-actual arguments to obtain only actual arguments and removes those potential, non-actual arguments that cannot be turned into actual arguments, as well as pairs in Att that refer to arguments no longer existing. Note that, by definition of structured X-dispute derivation, for some $A \subseteq \mathcal{A}$ and $\alpha \in \mathcal{L}$, $l : A \vdash_{\{\}} \alpha \in Args^a$ and $(l, \emptyset) \in Att^a$ necessarily.

For Example 6.2, $Actual(Args, Att)$ has

- $Args^a = \{l_1 : \{a, a\} \vdash_{\{\}} p, l_5 : \{f\} \vdash_{\{\}} f, l_3 : \{a\} \vdash_{\{\}} u, l^{**}_2 : \{c\} \vdash_{\{\}} q,$
  $l'''_{2\ \langle\{t\},\{d\}\rangle} : \{c, d\} \vdash_{\{\}} q, l'''_{2\ \langle\{t\},\{e\}\rangle} : \{c, e\} \vdash_{\{\}} q\}$;
- $Att^a = \{l_1 \rightsquigarrow \emptyset, l^{**}_2 \rightsquigarrow l_1, l_3 \rightsquigarrow l^{**}_2, l'''_{2\ \langle\{t\},\{d\}\rangle} \rightsquigarrow l_1, l'''_{2\ \langle\{t\},\{e\}\rangle} \rightsquigarrow l_1\}$.

Note that the potential argument labelled l'_2 does not contribute to $Actual(Args, Att)$, as this is a potential argument that cannot be turned into any actual argument.

It is easy to see that in the case of a *patient* selection function [13], namely a selection function *sel* such that, for any $S \subseteq \mathcal{L}$, $sel(S) \in \mathcal{A}$ iff $S - \mathcal{A} = \{\}$, the actual dialectical structure coincides with the originally computed dialectical structure:

Proposition 7.1. *Let $(Args, Att)$ be the dialectical structure computed by a structured X-dispute derivation for some sentence w.r.t. a patient selection function sel (and any choices of the other parameters). Then, $Actual(Args, Att) = (Args, Att)$.*

Note that the selection function used in Example 6.1 is patient whereas the selection function used in Example 6.2 is not, as, for example, at step 1 it selects a in the unmarked part of the potential argument labelled l_1 even though $u \notin \mathcal{A}$ could be selected there. In line with Proposition 7.1, $Actual(Args, Att) = (Args, Att)$ in Example 6.1.

7.2.2. Pruned dialectical forest and grounded semantics

$Actual(Args, Att) = (Args^a, Att^a)$ corresponds, in general, to a set of trees, that we refer to as (*dialectical*) *forest* and denote $\mathcal{F}(Args^a, Att^a)$. One of these trees has as root the potential argument in $Args^a$ with label l such that $l \rightsquigarrow \emptyset \in Att^a$ (and

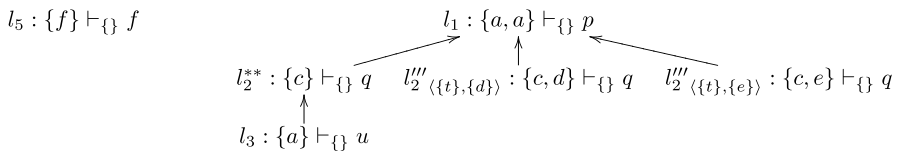


Fig. 9. Dialectical forest $\mathcal{F}(Args^a, Att^a)$ for Example 6.2.

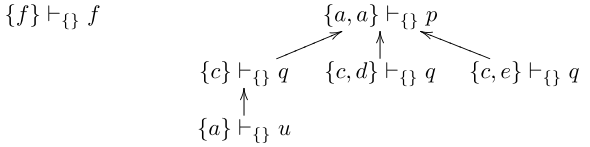


Fig. 10. Pruned dialectical forest $\mathcal{F}^p(Args^a, Att^a)$ for Example 6.2.

Att). The other trees have as roots actual arguments that attack no argument in the actual dialectical structure, because the potential arguments they attacked in the computed dialectical structure did not result in any arguments in the actual dialectical structure. Formally:

Definition 7.4. Let $(Args, Att)$ be the dialectical structure computed by a structured X-dispute derivation for some sentence. The (*dialectical*) *forest* $\mathcal{F}(Args^a, Att^a)$ obtained from $(Args^a, Att^a) = Actual(Args, Att)$ is the set of all trees $\mathcal{T}$ such that

- the root of $\mathcal{T}$ is
  - either the (actual) argument in $Args^a$ with label l such that $l \rightsquigarrow \emptyset \in Att^a$,
  - or an (actual) argument $l : A \vdash_{\{\}} \sigma \in Args^a$ such that there exists no $l \rightsquigarrow l' \in Att^a$;
- if a node in $\mathcal{T}$ is an argument in $Args^a$ with label l_N, then the node has as children all the arguments in $Args^a$ with label l_M such that $l_M \rightsquigarrow l_N \in Att^a$.

The *pruned* (*dialectical*) *forest* $\mathcal{F}^p(Args^a, Att^a)$ is the dialectical forest $\mathcal{F}(Args^a, Att^a)$ without the labels.

In Example 6.2, the forest consists of two trees, given in Fig. 9. The pruned forest is given in Fig. 10.

It is easy to see that, if a patient selection function is used, then the dialectical forest consists of a single tree, and this is the $\mathcal{T}^*$ given earlier:

Proposition 7.2. *Let* $(Args, Att)$ *be the dialectical structure computed by a structured X-dispute derivation for some sentence w.r.t. a patient selection function sel* (*and any choices of the other parameters*)*, and* $(Args^a, Att^a) = Actual(Args, Att)$*. Then* $\mathcal{F}(Args^a, Att^a) = \{\mathcal{T}^*(Args, Att)\}$.

In line with Proposition 7.2, for Example 6.1 the forest consists of a single tree.

Since actual arguments correspond to ABA arguments, in the remainder of the paper we will abuse notation and use actual arguments $A \vdash_{\{\}} \sigma$, labelled actual arguments $l : A \vdash_{\{\}} \sigma$ and the corresponding ABA arguments for σ supported by A interchangeably. Thus, for example, we may say that an ABA argument $\mathtt{b}$ attacks $l : A \vdash_{\{\}} \sigma$ to mean that $\mathtt{b}$ attacks the ABA argument corresponding to $l : A \vdash_{\{\}} \sigma$. Also, under this convention, the nodes in trees in pruned forests are all ABA arguments.

We can now prove soundness of structured X-dispute derivations as far as the computed dialectical structure is concerned, by proving correspondences between trees in the pruned forest and the dispute trees of [13,14].

The pruned dialectical forest obtained for GB-choices of parameters is a set of grounded dispute trees, where odd-level (even-level) nodes have the proponent (opponent, respectively) status.[8] For example, the pruned dialectical forest obtained for Example 6.1 (consisting of the single tree in Fig. 8 (left) after removing the labels) consists of a single grounded dispute tree. More generally and formally:

Theorem 7.2 *(Soundness of structured X-dispute derivations w.r.t. grounded semantics – dialectical structure). Let* $(Args, Att)$ *be the dialectical structure computed by a structured X-dispute derivation for a sentence* δ *w.r.t. GB-choices of parameters, and let* $(Args^a, Att^a) = Actual(Args, Att)$*. Then,*

(A) *every tree in the pruned forest* $\mathcal{F}^p(Args^a, Att^a)$ *is a grounded dispute tree* (*for the argument in its root*);

[8] We assume that the root of a tree is of level 1, and the level of a non-root node is the level of its parent +1.

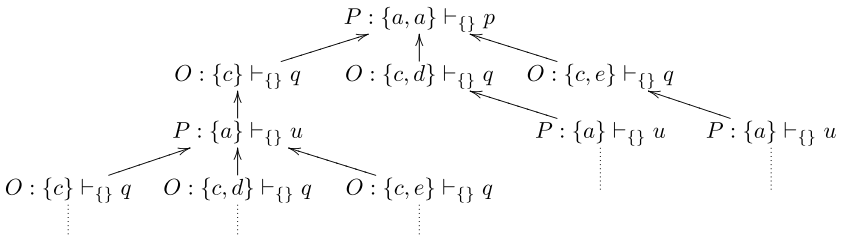


Fig. 11. Dispute tree obtained from the right-most tree in Fig. 10.

(B) *there exists a grounded dispute tree in the pruned forest* $\mathcal{F}^p(Args^a, Att^a)$ *for an argument for* δ.

The proof of this theorem is in Appendix B.3.

7.2.3. Expanded dialectical forest and admissible/ideal semantics

The pruned dialectical forest obtained for AB- or IB-choices of parameters may not be a set of admissible or ideal, respectively, dispute trees, because some of the trees in this forest may not be dispute trees in the first place. For example, the right-hand tree in the pruned forest in Fig. 10 is not a dispute tree, as:

- it does not fulfil condition 4 of the definition of dispute tree in that the middle and right-most leaves in this tree (originating from then potential argument labelled l_2''') must necessarily have the status of opponent nodes but they (incorrectly) have no children, and
- it does not fulfil condition 3 of the definition of dispute trees, since the left-most (proponent) leaf holds an argument that is attacked by three arguments (for q supported by $\{c\}$, $\{c, d\}$ and $\{c, e\}$ respectively) but there are no children of this node.

The absence of some nodes in the tree is caused by the deployment of f_{DbyD} and f_{CbyC} according to the AB- and IB-choices. However, it is easy to see that in this example a dispute tree can be obtained from the right-most tree in Fig. 10 by adding arguments to the tree, as sketched in Fig. 11. This dispute tree is infinite. All nodes added to it already occur in the given forest (in this case in the given tree itself). This expansion is defined formally below, making use of the following Lemma 7.1. Here, we use the following terminology: an argument is *attackable* if the set of arguments that attack it in the underlying ABA framework is non-empty. Note that if the support of an argument is empty then the argument is not attackable.

Lemma 7.1. *Let* $(Args, Att)$ *be the dialectical structure computed by a structured X-dispute derivation w.r.t. AB- or IB-choices of parameters, and let* $(Args^a, Att^a) = Actual(Args, Att)$. *Then, for every leaf N of (any tree in)* $\mathcal{F}^p(Args^a, Att^a)$ *holding an* attackable *argument* $A \vdash_{\{\}} \sigma_A$ *there exists a node M in (some tree in)* $\mathcal{F}^p(Args^a, Att^a)$ *holding an argument* $B \vdash_{\{\}} \sigma_B$ *such that* $\sigma_B = \overline{\alpha}$ *for some* $\alpha \in A$ *and M is even-level (odd-level) if N is odd-level (even-level, respectively).*

The proof of this lemma is in Appendix B.4. We will refer to any $B \vdash_{\{\}} \sigma_B$ as in Lemma 7.1 as $arg_{\mathcal{F}}(\alpha)$. This lemma guarantees that the following definition is well-formed:

Definition 7.5. Let $(Args, Att)$ be the dialectical structure computed by a structured X-dispute derivation w.r.t. AB- or IB-choices of parameters, and let $(Args^a, Att^a) = Actual(Args, Att)$. Let S and C be the support and culprits, respectively, computed by the same structured X-dispute derivation. Given $\mathcal{T} \in \mathcal{F}^p(Args^a, Att^a)$, let us construct a (possibly infinite) sequence $\mathcal{T}_0, \ldots, \mathcal{T}_n, \ldots$ of trees such that

- $\mathcal{T}_0 = \mathcal{T}$;
- suppose $\mathcal{T}_i$, for $i \geqslant 0$, has been constructed; then $\mathcal{T}_{i+1}$ is obtained by adding simultaneously to all leaves N of $\mathcal{T}_i$ holding an *attackable* argument $A \vdash_{\{\}} \sigma$:
  - all children $arg_{\mathcal{F}}(\alpha)$, for all $\alpha \in A \cap S$, if N is an odd-level node;
  - a child $arg_{\mathcal{F}}(\alpha)$, for some $\alpha \in A \cap C$, if N is an even-level node.

Then, the *expanded* (*dialectical*) *tree of* $\mathcal{T}$ is the limit of this sequence. Moreover, the *expanded* (*dialectical*) *forest of* $\mathcal{F}^p(Args^a, Att^a)$ is the set of all expanded trees of trees in $\mathcal{F}^p(Args^a, Att^a)$.

Given the right-most tree $\mathcal{T}$ in the forest in Fig. 10, Fig. 11 without the dotted lines shows $\mathcal{T}_1$ in the construction of the expanded tree of $\mathcal{T}$.

Theorem 7.3 *(Soundness of structured X-dispute derivations w.r.t. admissible/ideal semantics – dialectical structure). Let* $(Args, Att)$ *be the dialectical structure computed by a structured X-dispute derivation for a sentence* δ *w.r.t. AB-* (*or IB-*)*choices of parameters. Let* $\mathcal{F}^p(Args^a, Att^a)$ *be the pruned forest obtained from* $(Args^a, Att^a) = Actual(Args, Att)$*. Then,*

(A) *every tree in the expanded forest of* $\mathcal{F}^p(Args^a, Att^a)$ *is an admissible* (*ideal, respectively*) *dispute tree* (*for the argument in its root*);
(B) *there exists an admissible* (*ideal, respectively*) *dispute tree in the expanded forest of* $\mathcal{F}^p(Args^a, Att^a)$ *for an argument for* δ*.*

The proof of this theorem is in Appendix B.5.

Note that, by virtue of this theorem, each tree in the pruned forest $\mathcal{F}^p(Args^a, Att^a)$ can be seen as a finite representation of a possibly infinite admissible/ideal dispute tree.

8. Completeness results for X-dispute derivations and structured X-dispute derivations

We obtain completeness results in the case of p-acyclic ABA frameworks [14] (see Section 2) with a finite underlying language.

Theorem 8.1 *(Completeness of X-dispute derivations). Given a p-acyclic ABA framework* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ *with a finite* $\mathcal{L}$*, and a sentence* $\delta \in \mathcal{L}$*, if there exists an argument* $\mathtt{a}$ *for* δ *supported by* Σ *and a grounded/admissible/ ideal set* $\mathtt{A}$ *of arguments such that* $\mathtt{a} \in \mathtt{A}$ *then there exists a X-dispute derivation of support* Δ *for* δ *w.r.t. GB-/AB-/IB-choices of parameters* (*respectively*) *such that*

- $\Sigma \subseteq \Delta$*, and*
- $\Delta \subseteq Asm(\mathtt{A})$*, where* $Asm(\mathtt{A})$ *is the union of all sets of assumptions supporting arguments in* $\mathtt{A}$*.*

The proof of this theorem is in C.1.

Directly from Theorem 8.1 and from Theorem 7.1:

Corollary 8.1 *(Completeness of structured X-dispute derivations – support). Given a p-acyclic ABA framework* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ *with a finite* $\mathcal{L}$*, and a sentence* $\delta \in \mathcal{L}$*, if there exists an argument* $\mathtt{a}$ *for* δ *with support* Σ *and a grounded/admissible/ideal set* $\mathtt{A}$ *of arguments such that* $\mathtt{a} \in \mathtt{A}$ *then there exists a structured X-dispute derivation of support* Δ *for* δ *w.r.t. GB-/AB-/IB-choices of parameters* (*respectively*) *such that*

- $\Sigma \subseteq \Delta$*, and*
- $\Delta \subseteq Asm(\mathtt{A})$*, where* $Asm(\mathtt{A})$ *is the union of all sets of assumptions supporting arguments in* $\mathtt{A}$*.*

Theorem 8.2 *(Completeness of structured X-dispute derivations w.r.t. grounded semantics – dialectical structure). Given a p-acyclic ABA framework* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ *with a finite* $\mathcal{L}$*, and a sentence* $\delta \in \mathcal{L}$*, if there exists an argument* $\mathtt{a}$ *for* δ *with support* Σ *and a grounded dispute tree* $\mathcal{T}$ *with root* $\mathtt{a}$ *then there exists a structured X-dispute derivation of dialectical structure* $(Args, Att)$ *for* δ *w.r.t. GB-choices of parameters such that* $\mathcal{T}^*(Args, Att) = \mathcal{T}$*.*

The proof of this theorem is in Appendix C.2.

Theorem 8.3 *(Completeness of structured X-dispute derivations w.r.t. admissible/ideal semantics – dialectical structure). Given a p-acyclic ABA framework* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ *with a finite* $\mathcal{L}$*, and a sentence* $\delta \in \mathcal{L}$*, if there exists an argument* $\mathtt{a}$ *for* δ *with support* Σ *and an admissible/ideal dispute tree* $\mathcal{T}$ *with root* $\mathtt{a}$ *then there exists a structured X-dispute derivation of dialectical structure* $(Args, Att)$ *for* δ *w.r.t. AB-/IB-choices of parameters* (*respectively*) *such that the expanded forest of* $\mathcal{F}^p(Args^a, Att^a)$ *is* $\{\mathcal{T}'\}$*,* $\mathcal{T}'$ *is an admissible/ideal dispute tree for* $\mathtt{a}$ (*respectively*)*, and the argument defence set* $\mathtt{D}'$ *of* $\mathcal{T}'$ *is such that* $\mathtt{D}' \subseteq \mathtt{D}$*, where* $\mathtt{D}$ *is the argument defence set of* $\mathcal{T}$*.*

The proof of this theorem is in Appendix C.3. The following example shows a case with $\mathtt{D}' \subset \mathtt{D}$.

Example 8.1. Consider the ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ with

- $\mathcal{R} = \{p \leftarrow a; q \leftarrow b; q \leftarrow z; z \leftarrow a, b; r \leftarrow c; r \leftarrow d\}$;
- $\mathcal{A} = \{a, b, c, d\}$;
- $\bar{a} = q$, $\bar{b} = r$, $\bar{c} = \bar{d} = s$.

Consider the following arguments:

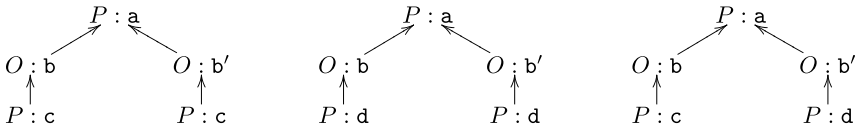


Fig. 12. Three admissible dispute trees for a in Example 8.1.

$Step$	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	$Args$	Att
0	$\{l_1 : \{\} \vdash_{\{p\}} p \rightsquigarrow \emptyset\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{l_1 : \{\} \vdash_{\{a\}} p \rightsquigarrow \emptyset\}$	$\{\}$	$\{a\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
2	$\{\}$	$\{l_2 : \{\} \vdash_{\{q\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p\}$	$\{l_1 \rightsquigarrow \emptyset\}$
3	$\{\}$	$\{l'_2 : \{\} \vdash_{\{b\}} q \rightsquigarrow l_1,$ $\{l''_2 : \{\} \vdash_{\{z\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p\}$	$\{l_1 \rightsquigarrow \emptyset\}$
4	$\{\}$	$\{l'_2 : \{\} \vdash_{\{b\}} q \rightsquigarrow l_1,$ $\{l^{**}_2 : \{\} \vdash_{\{a,b\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p\}$	$\{l_1 \rightsquigarrow \emptyset\}$
5	$\{l_3 : \{\} \vdash_{\{r\}} r \rightsquigarrow l'_2\}$	$\{l^{**}_2 : \{\} \vdash_{\{a,b\}} q \rightsquigarrow l_1\}$	$\{a\}$	$\{b\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1\}$
6	$\{l_3 : \{\} \vdash_{\{c\}} r \rightsquigarrow l'_2\}$	$\{l^{**}_2 : \{\} \vdash_{\{a,b\}} q \rightsquigarrow l_1\}$	$\{a, c\}$	$\{b\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1\}$
7	$\{\}$	$\{l^{**}_2 : \{\} \vdash_{\{a,b\}} q \rightsquigarrow l_1,$ $l_4 : \{\} \vdash_{\{s\}} s \rightsquigarrow l_3\}$	$\{a, c\}$	$\{b\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q,$ $l_3 : \{c\} \vdash_{\{\}} r\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1,$ $l_3 \rightsquigarrow l'_2\}$
8	$\{\}$	$\{l_4 : \{\} \vdash_{\{s\}} s \rightsquigarrow l_3\}$	$\{a, c\}$	$\{b\}$	$\{\}$	$Args_8$	Att_8
9	$\{\}$	$\{\}$	$\{a, c\}$	$\{b\}$	$\{\}$	$Args_8$	Att_8

Fig. 13. A structured X-dispute derivation for AB-choices of parameters for Example 12. At step 8, the potential argument labelled l^{**}_2 is moved into the *Args* component, by case 2(i)(b), as $sel(\{a, b\}) = b$ and $f_{CbyC}(\{b\}, C_8)$ holds (for AB-choices of parameters).

a: for p supported by $\{a\}$

b: for q supported by $\{b\}$

b′: for q supported by $\{a, b\}$

c: for r supported by $\{c\}$

d: for r supported by $\{d\}$

Then, all trees in Fig. 12 are admissible dispute trees for a, with argument defence set $D_1 = \{a, c\}$ (left tree, $\mathcal{T}_1$), $D_2 = \{a, d\}$ (middle tree, $\mathcal{T}_2$), and $D_3 = \{a, c, d\}$ (right tree, $\mathcal{T}_3$). A possible structured X-dispute derivations for p, w.r.t. AB-choices of parameters, is given in Fig. 13, with computed dialectical structure $(Args_8, Att_8)$ where

$$Args_8 = \left\{l_1 : \{a\} \vdash_{\{\}} p, l'_2 : \{b\} \vdash_{\{\}} q, l^{**}_2 : \{b\} \vdash_{\{a\}} q, l_3 : \{c\} \vdash_{\{\}} r\right\} \quad \text{and}$$

$$Att_8 = \left\{l_1 \rightsquigarrow \emptyset, l'_2 \rightsquigarrow l_1, l^{**}_2 \rightsquigarrow l_1, l_3 \rightsquigarrow l'_2\right\}$$

and with expanded forest $\{\mathcal{T}_1\}$. Another structured X-dispute derivations for p, for the same AB-choices of parameters, is obtained by replacing, in Fig. 13, c by d in any potential argument and in D, and has expanded forest $\{\mathcal{T}_2\}$. No structured X-dispute derivation is possible, for AB-choices of parameters, with resulting expanded forest $\{\mathcal{T}_3\}$, due to the definition of f_{CbyC} for AB-choices of parameters. Since $D_1, D_2 \subset D_3$, Theorem 8.3 holds nonetheless. Similarly for IB-choices of parameters (except that these would give $\mathcal{F}_5 = \mathcal{F}_6 = \mathcal{F}_7 = \{\{b\}\}$ and $\mathcal{F}_8 = \mathcal{F}_9 = \{\{b\}, \{a, b\}\}$, and two more steps in the derivation to ascertain that $Fail(\{b\})$ and $Fail(\{a, b\})$ hold). Finally, note that, for GB-choices of parameters, it is possible to construct a structured X-dispute derivation with resulting expanded forest $\{\mathcal{T}_3\}$, e.g. given by the derivation in Fig. 13 till step 7, and then

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	$\mathcal{F}$	*Args*	*Att*
$8'$	$\{l_5 : \{\} \vdash_{\{r\}} r \rightsquigarrow l_2^{**}\}$	$\{l_4 : \{\} \vdash_{\{s\}} s \rightsquigarrow l_3\}$	$\{a, c\}$	$\{b, b\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q,$ $l_3 : \{c\} \vdash_{\{\}} r,$ $l_2^{**} : \{b\} \vdash_{\{a\}} q\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1,$ $l_3 \rightsquigarrow l'_2\}$ $l_2^{**} \rightsquigarrow l_1\}$
$9'$	$\{l_5 : \{\} \vdash_{\{d\}} r \rightsquigarrow l_2^{**}\}$	$\{l_4 : \{\} \vdash_{\{s\}} s \rightsquigarrow l_3\}$	$\{a, c, d\}$	$\{b, b\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q,$ $l_3 : \{c\} \vdash_{\{\}} r,$ $l_2^{**} : \{b\} \vdash_{\{a\}} q\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1,$ $l_3 \rightsquigarrow l'_2\}$ $l_2^{**} \rightsquigarrow l_1\}$
10	$\{\}$	$\{l_4 : \{\} \vdash_{\{s\}} s \rightsquigarrow l_3,$ $l_6 : \{\} \vdash_{\{s\}} s \rightsquigarrow l_5\}$	$\{a, c, d\}$	$\{b, b\}$	$\{\}$	$\{l_1 : \{a\} \vdash_{\{\}} p,$ $l'_2 : \{b\} \vdash_{\{\}} q,$ $l_3 : \{c\} \vdash_{\{\}} r,$ $l_2^{**} : \{b\} \vdash_{\{a\}} q,$ $l_5 : \{d\} \vdash_{\{\}} r\}$	$\{l_1 \rightsquigarrow \emptyset,$ $l'_2 \rightsquigarrow l_1,$ $l_3 \rightsquigarrow l'_2\}$ $l_2^{**} \rightsquigarrow l_1,$ $l_5 \rightsquigarrow l_2^{**}\}$
11	$\{\}$	$\{l_4 : \{\} \vdash_{\{s\}} s \rightsquigarrow l_3\}$	$\{a, c, d\}$	$\{b, b\}$	$\{\}$	$Args_{10}$	Att_{10}
12	$\{\}$	$\{\}$	$\{a, c, d\}$	$\{b, b\}$	$\{\}$	$Args_{10}$	Att_{10}

Note that the restriction to p-acyclicity for the completeness results amounts to requiring that it is possible to compute arguments or fail to compute arguments finitely. This is important since our dispute derivations compute arguments top-down (from the root of argument trees to the leaves). In order to drop this restriction, our notion of (structured) X-dispute derivations need to be extended by some form of loop-checking, to detect finitely an infinite failure to compute arguments.

9. Results for other argumentation semantics

In this section we discuss how the soundness and completeness results given in earlier sections for structured X-dispute derivations extend to other argumentation semantics. Note that, by virtue of Theorem 7.1, these results, where applicable, also hold for X-dispute derivations.

Structured X-dispute derivations w.r.t. AB-choices of parameters are a sound and complete mechanism for the preferred semantics, as follows. As far as support is concerned, soundness is a direct corollary of Corollary 5.3 and Theorem 7.1:

Corollary 9.1 *(Soundness of structured X-dispute derivations w.r.t. preferred semantics – support). Given a structured X-dispute derivation of support $\Delta \subseteq \mathcal{A}$ and dialectical structure $(Args, Att)$ for $\delta \in \mathcal{L}$ w.r.t. AB-choices of parameters, there exists a preferred set of assumptions Δ^* such that*

- *$\Delta \subseteq \Delta^*$ and Δ^* is preferred*;
- *there exists $\Delta' \subseteq \Delta^*$ and an argument for δ supported by Δ'.*

Since every preferred set of assumptions/arguments is admissible, completeness as far as support is concerned is a direct corollary of Corollary 8.1:

Corollary 9.2 *(Completeness of structured X-dispute derivations w.r.t. preferred semantics – support). Given a p-acyclic ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ with a finite $\mathcal{L}$, and a sentence $\delta \in \mathcal{L}$, if there exists an argument $\mathtt{a}$ for δ with support Σ and a preferred set $\mathtt{A}$ of arguments such that $\mathtt{a} \in \mathtt{A}$ then there exists a structured X-dispute derivation of support Δ for δ w.r.t. AB-choices of parameters such that*

- *$\Sigma \subseteq \Delta$, and*
- *$\Delta \subseteq Asm(\mathtt{A})$, where $Asm(\mathtt{A})$ is the union of all sets of assumptions supporting arguments in $\mathtt{A}$.*

In order to formulate soundness and completeness results as far as the dialectical structure is concerned, we need to generalise the notion of admissible dispute tree to that of *preferred dispute forest*:

Definition 9.1. A set of admissible dispute trees is a *preferred dispute forest* iff the set of all arguments labelling proponent nodes in the trees is a preferred set of arguments.

Trivially, the union of all sets of assumptions supporting all arguments labelling proponent nodes in all the trees in a preferred dispute forest is a preferred set of assumptions (see Section 2). As an illustration, consider the ABA framework in Example 8.1 and let $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ be the trees (left-to-right) in Fig. 12. Since $\{a, c, d\}$ is the only preferred set of assumptions in this example, $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$, $\{\mathcal{T}_1, \mathcal{T}_2\}$ and $\{\mathcal{T}_1, \mathcal{T}_3\}$ are preferred dispute forests (note that there are other preferred dispute forests in this example).

Trivially, directly from Theorem 4.4 in [3] (that every admissible set of assumptions is contained in a preferred set) and by Theorem 3.2 in [14] (that for every argument in an admissible set there is an admissible dispute tree for that argument), it follows that for every argument in a preferred set there is an admissible dispute tree for that argument in a preferred dispute forest, and, conversely, if there is an admissible dispute tree for an argument in a preferred dispute forest then that argument is in a preferred set. Then, the following results for structured X-dispute derivations w.r.t. AB-choices of parameters, as far as the dialectical structure is concerned, are direct corollaries of Theorems 7.3 and 8.3 respectively:

Corollary 9.3 *(Soundness of structured X-dispute derivations w.r.t. preferred semantics – dialectical structure). Let* $(Args, Att)$ *be the dialectical structure computed by a structured X-dispute derivation for a sentence* δ *w.r.t. AB-choices of parameters. Let* $\mathcal{F}$ *be the expanded forest of* $\mathcal{F}^p(Args^a, Att^a)$*, the pruned forest obtained from* $(Args^a, Att^a) = Actual(Args, Att)$*. Then,*

(A) *there exists a preferred dispute forest* $\mathcal{F}^*$ *such that* $\mathcal{F} \subseteq \mathcal{F}^*$*, and*
(B) *there exists an admissible dispute tree in* $\mathcal{F}$ *for an argument for* δ*.*

Corollary 9.4 *(Completeness of structured X-dispute derivations w.r.t. preferred semantics – dialectical structure). Given a p-acyclic ABA framework* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ *with a finite* $\mathcal{L}$*, and a sentence* $\delta \in \mathcal{L}$*, if there exists an argument* a *for* δ *with support* Σ *and a preferred dispute forest* $\mathcal{F}$ *with a tree with root* a *in* $\mathcal{F}$ *then there exists a structured X-dispute derivation of dialectical structure* $(Args, Att)$ *for* δ *w.r.t. AB-choices of parameters, with* $\mathcal{F}^*$ *the expanded forest of* $\mathcal{F}^p(Args^a, Att^a)$*, such that* $\mathcal{F}^* \subseteq \mathcal{F}$*.*

Let us now briefly consider the complete semantics (see Section 2). Since every preferred set of assumptions is complete (Corollary 5.1 in [3]), and, as a consequence, every admissible set of assumptions is contained in a complete set, soundness and completeness results for structured X-dispute derivations w.r.t. AB-choices of parameters under the complete semantics can also be given. Moreover, since the grounded set of assumptions is complete, trivially structured X-dispute derivations w.r.t. GB-choices of parameters are a sound (but incomplete) mechanism under the complete semantics.

10. Related work

Our definition of X-dispute derivation is a generalisation of the AB-dispute derivations of [13,14] and the GB- and IB-dispute derivations of [14], in that these derivations can be obtained by instantiating our X-dispute derivations. Implementations of AB-, GB- and IB-dispute derivations can be obtained from implementing X-dispute derivations, by providing appropriate implementations of the parameters underlying X-dispute derivations, e.g. as in [9]. A corollary of our results, extending the results in [14], is completeness (for the p-acyclic ABA frameworks of [14]) of GB-dispute derivations.

Our definition of structured X-dispute derivation bares some similarities with the notion of *structured AB-dispute derivation* of [25,26], for computation under the admissible semantics. However, the dialectical structure computed by the latter differs from the one we compute in the instance of structured X-dispute derivations for AB-choices of parameters in that the former arbitrarily "hangs" below any opponent argument added in (the equivalent of our) cases 2(i)(b) and 2(ii), after f_{CbyC} succeeds, an argument already in the dialectical structure or currently being constructed by the proponent. Moreover, no completeness results are proven for the structured AB-dispute derivation of [26]. Also, the notion of structured AB-dispute derivation of [25] is only defined for patient selection functions. A variant of the structured AB-dispute derivations of [26] for computation under the grounded semantics is sketched in [24]. This differs from the instance of our structured X-dispute derivations for GB-choices of parameters in the same way as the structured AB-dispute derivation of [26] from our instance for AB-choices of parameters. Moreover, no formal results for this variant have been proven. Our instance for IB-choices of parameters of structured X-dispute derivations is completely novel.

Kakas and Toni [44,35] also developed argumentation-theoretic proof procedures for the admissibility and grounded semantics (by suitably varying parameters, loosely speaking corresponding to our filtering parameters), as well as the weak stability [33] and the acceptability [34] argumentation semantics. Their proof procedures operate on a form of dispute trees but are defined only for logic programs. Moreover, these procedures do not consider the ideal semantics.

DeLP [27] is also a logic-programming-based approach, supporting argumentation with sets of defeasible and strict rules and incorporating reasoning with specificity. The DeLP system is based upon a notion of dialectical trees, incrementally constructed and used to determine whether a given query is 'warranted' (and thus positively answered), 'unwarranted' (and thus negatively answered), or neither 'warranted' nor 'unwarranted' (and thus undecided; a query can also be unknown, if not in the given language). ABA admits instances and variants for reasoning with defeasible and strict rules and preferences [37,43] while at the same time being an instance of abstract argumentation [12] with a clear distinction between semantics and computation. Differently from DeLP, (structured) X-dispute derivations focus on answering queries positively, in that they incorporate a successful strategy for conducting a dispute. However, (structured) X-dispute derivations return a justification (in the form of a support set and, for structured X-dispute derivations, a dialectical structure), lacking in DeLP except for some recent attempts [28].

The notion of *dispute derivations* from [13,14] that we extend here is also the starting point for the computational framework of [16,42], but there this notion is used to support computation, under several semantics (admissible, grounded, ideal and sceptically preferred), in abstract argumentation. That notion is at the same time a simplification (from ABA

to abstract argumentation) and a generalisation (to deal with the sceptically preferred semantics and apply the grounded semantics more broadly) of the method of [14] that is also a starting point for our (structured) X-dispute derivations. We have focused on providing a sound and complete computational method for ABA (thus incorporating also computation of arguments and attacks, in addition to dispute trees, and exploiting the fact that different arguments may share the same assumptions) which, in addition, is fully parameterised, to aid transparent and modular implementations as well as broad experimentation in support of diverse applications (e.g. as preliminarily explored in [9]).

Several computational models for abstract argumentation have been proposed, as reviewed in [39] and [45]. These fall into three categories: 1) methods based upon proponent-opponent games for the construction of acceptable trees, broadly speaking based upon the approach of [32], e.g. the aforementioned [16,42] as well as several others ([19,46,7], to mention just a few); 2) methods based on a labelling algorithms to determine which arguments are IN, OUT or UNDECIDED (e.g. [6]); and 3) methods for computing full extensions using answer set programming techniques (e.g. [22], see [45] for a survey of these approaches). Our (structured) X-dispute derivations fall into the first category, but are defined for ABA as discussed earlier (in comparison with [16,42]). Since ABA frameworks can be mapped onto abstract argumentation frameworks, as proven in [14], one could in principle apply any of the methods 1)–3) to ABA. However, note that these methods for abstract argumentation apply to *finite* argumentation frameworks only, whereas our (structured) X-dispute derivations (as well as AB-, GB-, and IB-dispute derivations) can be applied to any ABA frameworks, even when their corresponding abstract argumentation framework is infinite.

Bryant et al. [4,5] give an argumentation engine, implemented in a Java-based form of Prolog, for detecting whether a given input is admissible, in the context of a precursor of the argumentation framework of [40], and based upon the computational model of [32]. Along the same lines, South et al. [41] propose a flexible Java argument engine and API, but to assess whether an input is admissible/preferred or grounded. Our structured X-dispute derivations are also inspired by [32], in that they can be seen as games between a proponent and an opponent. However, they are constructive methods for computing "acceptable" trees and arguments/set of assumptions, incrementally, and allowing (depending on the choices of parameters) for the interleaving of the construction of dispute trees and of arguments and attacks (and exploiting the fact that different arguments may share the same assumptions). Moreover, we have proven soundness and correctness results for (structured) X-dispute derivations w.r.t. three different notions of "acceptability". Also, our structured X-dispute derivations are a formal model, decoupled from any implementation, that can result into several prototype implementations.

Efstathiou and Hunter [21] propose algorithms for supporting argumentation in propositional logic, following the method of [2]. In particular, they use connection graphs [36] and resolution for generating arguments and counter-arguments, in the form of canonical undercuts [2], as well as trees with propositional logic arguments. These trees are similar in spirit to the trees underlying DeLP, and differ from the dispute trees underpinning our approach in that they do not correspond to abstract argumentation semantics. Our focus has been on the interleaving of the construction of (admissible, grounded, ideal) dispute trees and of arguments and attacks (and exploiting the fact that different arguments may share the same assumptions), whereas their focus is on the efficient computation of arguments and counter-arguments.

11. Conclusions

We have presented a notion of structured X-dispute derivations for ABA, generalising, and admitting as special instances, GB-, AB- and IB-dispute derivations [13,14] as well as, in the logic programming instance of ABA, SLDNF and the abductive proof procedure of [23]. We have defined structured X-dispute derivations as an extension of X-dispute derivations, in turn generalising, and admitting as special instances, GB-, AB- and IB-dispute derivations as well as, in the logic programming instance of ABA, SLDNF and the abductive proof procedure of [23].

Both X-dispute derivations and structured X-dispute derivations single out explicitly design choices underlying, and implicit in, GB-, AB- and IB-dispute derivations, and pave the way to a unified, modular implementation of these mechanisms. In the logic programming instance, they go beyond the existing procedures by supporting, in particular, query answering under the ideal semantics for logic programming [1], corresponding to the ideal semantics for the logic programming instance of ABA [14]. Additionally, structured X-dispute derivations compute the dialectical structure (of arguments and counter-arguments) providing a justification for the given query (claim/conclusion), that is useful (and, arguably, essential) for "explaining" queries. Structured X-dispute derivations thus provide a novel mechanism to support justified query answering in all instances of ABA, including logic programming.

We have proven soundness and completeness results for X-dispute derivations and structured X-dispute derivations, for specific choices of parameters, w.r.t. grounded, admissible and ideal semantics.

There are several directions that future work may take.

It would be useful to explore further cross-fertilisation with logic programming, for example to introduce further mechanisms of filtering to guarantee completeness for non-p-acyclic ABA frameworks, in the case of the computation of the grounded semantics, as done for the computation of the well-founded semantics in [8].

Also, it would be interesting to see how (structured) X-dispute derivations could be extended to support the computation of the sceptically preferred semantics for ABA (whereby a conclusion is held if it is supported by all preferred sets of assumptions), e.g. tailoring to ABA the approach of [16,42] for abstract argumentation.

(Structured) X-dispute derivations focus on answering queries positively, in that they incorporate a successful strategy for conducting a dispute. It would be useful to extend these notions to provide justifications for queries that cannot be answered positively (namely for which a successful strategy does not exist).

Further, in order to support (existing and new) applications of ABA, such as the ones described in [15], it would be essential to provide a modular implementation of structured X-dispute derivations, with appropriate graphical user interfaces to instantiate the parameters and visualise the computed dialectical structure as well as the "debate" leading to its construction.

In addition to subsuming and extending previous proof theories for ABA, the general framework of structured X-dispute derivations offers opportunities for developing a variety of argumentation systems. For example, the experiments in [9] rely upon an implementation, over a parallel platform, of a variant of the instance of our structured X-dispute derivations for AB-choices of parameters. The implementation explores, in parallel over a multi-core platform, different realisations of the implementation choice parameters and selection function, with beneficial performance effects. Moreover, the experimentation shows that, in general, it is not possible to commit to any specific choice of parameters without risking a computational explosion, thus further justifying the non-committal approach to these parameters adopted in this paper. Implementation and experimentation for this prototype are empowered by our parameterisation here. It would be interesting to further this experimentation for GB- and IB-choices of parameters, and over different parallel architectures, such as the cloud. Moreover, it would be interesting to explore whether useful heuristics can be drawn, for ABA frameworks with specific "structural" characteristics (e.g. in terms of the maximum number of rules for the contrary of assumptions), as to which variant of our approach is computationally more feasible.

Acknowledgements

We thank the anonymous referees for their comments, which we believe helped improve this paper considerably. We also thank Adrian Hadad for proof-reading the paper (and checking the examples in particular), and Rob Craven for useful feedback on the flowcharts.

Appendix A. GB-, AB-, IB-dispute derivations

The definitions in this appendix are from [13,14], but adopting the convention, when defining changes in tuples over dispute derivations, that omitted elements are unchanged.

Definition A.1. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ be an ABA framework. Given a selection function, a *GB-dispute derivation of a defence set* Δ for a sentence δ is a finite sequence of quadruples

$$\langle \mathcal{P}_0, \mathcal{O}_0, D_0, C_0 \rangle, \quad \ldots, \quad \langle \mathcal{P}_i, \mathcal{O}_i, D_i, C_i \rangle, \quad \ldots, \quad \langle \mathcal{P}_n, \mathcal{O}_n, D_n, C_n \rangle$$

where

$$\mathcal{P}_0 = \{\delta\} \qquad D_0 = \mathcal{A} \cap \{\delta\} \qquad \mathcal{O}_0 = C_0 = \{\}$$
$$\mathcal{P}_n = \mathcal{O}_n = \{\} \qquad \Delta = D_n$$

and for every $0 \leqslant i < n$, only one σ in $\mathcal{P}_i$ or one S in $\mathcal{O}_i$ is selected, and:

1. If $\sigma \in \mathcal{P}_i$ is selected then
   (i) if σ is an assumption, then
   $$\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \quad \mathcal{O}_{i+1} = \mathcal{O}_i \cup \left\{\{\overline{\sigma}\}\right\}$$
   (ii) if σ is not an assumption, then there exists some inference rule $\sigma \leftarrow R \in \mathcal{R}$ such that $C_i \cap R = \{\}$ and
   $$\mathcal{P}_{i+1} = \left(\mathcal{P}_i - \{\sigma\}\right) \cup R \quad D_{i+1} = D_i \cup (\mathcal{A} \cap R)$$
2. If S is selected in $\mathcal{O}_i$ and σ is selected in S then
   (i) if σ is an assumption, then
   (a) either σ is ignored, i.e.
   $$\mathcal{O}_{i+1} = \left(\mathcal{O}_i - \{S\}\right) \cup \left\{S - \{\sigma\}\right\}$$
   (b) or $\sigma \notin D_i$ and
   $$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \quad \mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$$
   $$D_{i+1} = D_i \cup \left(\{\overline{\sigma}\} \cap \mathcal{A}\right) \quad C_{i+1} = C_i \cup \{\sigma\}$$
   (ii) if σ is not an assumption, then
   $$\mathcal{O}_{i+1} = \left(\mathcal{O}_i - \{S\}\right) \cup \left\{S - \{\sigma\} \cup R \,|\, \sigma \leftarrow R \in \mathcal{R}\right\}$$

Definition A.2. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ be an ABA framework. Given a selection function, an *AB-dispute derivation of a defence set* Δ for a sentence δ is a finite sequence of quadruples

$$\langle \mathcal{P}_0, \mathcal{O}_0, D_0, C_0 \rangle, \quad \ldots, \quad \langle \mathcal{P}_i, \mathcal{O}_i, D_i, C_i \rangle, \quad \ldots, \quad \langle \mathcal{P}_n, \mathcal{O}_n, D_n, C_n \rangle$$

where

$$\mathcal{P}_0 = \{\delta\} \qquad D_0 = \mathcal{A} \cap \{\delta\} \qquad \mathcal{O}_0 = C_0 = \{\}$$
$$\mathcal{P}_n = \mathcal{O}_n = \{\} \qquad \Delta = D_n$$

and for every $0 \leqslant i < n$, only one σ in $\mathcal{P}_i$ or one S in $\mathcal{O}_i$ is selected, and:

1. If $\sigma \in \mathcal{P}_i$ is selected then
   (i) if σ is an assumption, then
   $$\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \quad \mathcal{O}_{i+1} = \mathcal{O}_i \cup \left\{\{\overline{\sigma}\}\right\}$$
   (ii) if σ is not an assumption, then there exists some inference rule $\sigma \leftarrow R \in \mathcal{R}$ such that $C_i \cap R = \{\}$ and
   $$\mathcal{P}_{i+1} = \left(\mathcal{P}_i - \{\sigma\}\right) \cup (R - D_i)$$
   $$D_{i+1} = D_i \cup (\mathcal{A} \cap R)$$
2. If S is selected in $\mathcal{O}_i$ and σ is selected in S then
   (i) if σ is an assumption, then
   (a) either σ is ignored, i.e.
   $$\mathcal{O}_{i+1} = \left(\mathcal{O}_i - \{S\}\right) \cup \left\{S - \{\sigma\}\right\}$$
   (b) or $\sigma \notin D_i$ and $\sigma \in C_i$[9] and
   $$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$$
   (c) or $\sigma \notin D_i$ and $\sigma \notin C_i$[10] and
   (c.1) if $\overline{\sigma}$ is not an assumption, then
   $$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \quad \mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\} \quad C_{i+1} = C_i \cup \{\sigma\}$$
   (c.2) if $\overline{\sigma}$ is an assumption, then
   $$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \quad D_{i+1} = D_i \cup \{\overline{\sigma}\} \quad C_{i+1} = C_i \cup \{\sigma\}$$
   (ii) if σ is not an assumption, then
   $$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \left\{S - \{\sigma\} \cup R \mid \sigma \leftarrow R \in \mathcal{R} \text{ and } R \cap C_i = \{\}\right\}$$

In this paper we use the variant of AB-dispute derivations where case 2(i)(c) is:

(c) or $\sigma \notin D_i$ and $\sigma \notin C_i$ and

$$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \quad \mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$$
$$D_{i+1} = D_i \cup \left(\mathcal{A} \cap \{\overline{\sigma}\}\right) \quad C_{i+1} = C_i \cup \{\sigma\}$$

This variant of AB-dispute derivations eliminates a further form of filtering, given by case (c.1), resulting in only a modest performance improvement. It is easy to see that this variant is equivalent to the original AB-dispute derivations (see [26,24] for details).

The definition of IB-dispute derivation uses the marking mechanism described in Section 4, Notation 1, and is as follows:

Definition A.3. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ be an ABA framework. Given a selection function, an *IB-dispute derivation of an ideal support* Δ for a sentence δ is a finite sequence of tuples

$$\langle \mathcal{P}_0, \mathcal{O}_0, D_0, C_0, \mathcal{F}_0 \rangle, \quad \ldots, \quad \langle \mathcal{P}_i, \mathcal{O}_i, D_i, C_i, \mathcal{F}_i \rangle, \quad \ldots, \quad \langle \mathcal{P}_n, \mathcal{O}_n, D_n, C_n, \mathcal{F}_n \rangle$$

where

[9] This case is in [14] but not in [13].
[10] The condition $\sigma \notin C_i$ in case (c) and case (c.2) are in [14] but not in [13].

$$\mathcal{P}_0 = \{\delta\} \quad D_0 = \mathcal{A} \cap \{\delta\} \quad \mathcal{O}_0 = C_0 = \mathcal{F}_0 = \{\}$$
$$\mathcal{P}_n = \mathcal{O}_n = \mathcal{F}_n = \{\} \quad \Delta = D_n$$

and for every $0 \leqslant i < n$, only one σ in $\mathcal{P}_i$ or one S in $\mathcal{O}_i$ or one S in $\mathcal{F}_i$ is selected, and:

1. If $\sigma \in \mathcal{P}_i$ is selected then
   (i) if σ is an assumption, then
   $$\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \quad \mathcal{O}_{i+1} = \mathcal{O}_i \cup \left\{\{\overline{\sigma}\}\right\}$$
   (ii) if σ is not an assumption, then there exists some inference rule $\sigma \leftarrow R \in \mathcal{R}$ such that $C_i \cap R = \{\}$ and
   $$\mathcal{P}_{i+1} = \left(\mathcal{P}_i - \{\sigma\}\right) \cup (R - D_i) \quad D_{i+1} = D_i \cup (\mathcal{A} \cap R)$$
2. If S is selected in $\mathcal{O}_i$ and σ is selected in S_u then
   (i) if σ is an assumption, then
   (a) either σ is ignored, i.e.
   $$\mathcal{O}_{i+1} = \left(\mathcal{O}_i - \{S\}\right) \cup \left\{m(\sigma, S)\right\}$$
   (b) or $\sigma \notin D_i$ and $\sigma \in C_i$ and
   $$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \quad \mathcal{F}_{i+1} = \mathcal{F}_i \cup \left\{u(S)\right\}$$
   (c) or $\sigma \notin D_i$ and $\sigma \notin C_i$ and
   (c.1) if $\overline{\sigma}$ is not an assumption, then
   $$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \quad \mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$$
   $$C_{i+1} = C_i \cup \{\sigma\} \quad \mathcal{F}_{i+1} = \mathcal{F}_i \cup \left\{u(S)\right\}$$
   (c.2) if $\overline{\sigma}$ is an assumption, then
   $$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \quad D_{i+1} = D_i \cup \{\overline{\sigma}\}$$
   $$C_{i+1} = C_i \cup \{\sigma\} \quad \mathcal{F}_{i+1} = \mathcal{F}_i \cup \left\{u(S)\right\}$$
   (ii) if σ is not an assumption, then
   $$\mathcal{O}_{i+1} = \left(\mathcal{O}_i - \{S\}\right) \cup \left\{\left(S - \{\sigma\}\right) \cup R \mid \sigma \leftarrow R \in \mathcal{R} \text{ and } R \cap C_i = \{\}\right\}$$
   $$\mathcal{F}_{i+1} = \mathcal{F}_i \cup \left\{\left(S - \{\sigma\}\right) \cup R \mid \sigma \leftarrow R \in \mathcal{R} \text{ and } R \cap C_i \neq \{\}\right\}$$
3. If S is selected in $\mathcal{F}_i$ then *Fail*(S) and
   $$\mathcal{F}_{i+1} = \mathcal{F}_i - \{S\}$$

In this paper we use the variant of IB-dispute derivations where case 2(i)(c) is:

(c) or $\sigma \notin D_i$ and $\sigma \notin C_i$ and

$$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \quad \mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$$
$$D_{i+1} = D_i \cup \left(\mathcal{A} \cap \{\overline{\sigma}\}\right) \quad C_{i+1} = C_i \cup \{\sigma\} \quad \mathcal{F}_{i+1} = \mathcal{F}_i \cup \left\{u(S)\right\}$$

Similarly to the case of AB-dispute derivations, this variant is trivially equivalent to the original definition.

Appendix B. Proofs for Section 7

B.1. Proof of Theorem 7.1

We prove the theorem constructively, by mapping one kind of derivation onto the other kind, with the same support.

From a structured X-dispute derivation to a X-dispute derivation

We first show how to construct a X-dispute derivation X-dd:

$$\langle \mathcal{P}'_0, \mathcal{O}'_0, D_0, C_0, \mathcal{F}_0\rangle, \quad \ldots, \quad \langle \mathcal{P}'_n, \mathcal{O}'_n, D_n, C_n, \mathcal{F}_n\rangle$$

from a given structured X-dispute derivation sX-dd:

$$\langle \mathcal{P}_0, \mathcal{O}_0, D_0, C_0, \mathcal{F}_0, Args_0, Att_0\rangle, \quad \ldots, \quad \langle \mathcal{P}_n, \mathcal{O}_n, D_n, C_n, \mathcal{F}_n, Args_n, Att_n\rangle$$

This construction relies upon two mappings

$$m_P(\mathcal{X}) = \bigcup_{(l:S_m \vdash_{S_u} s \rightsquigarrow l') \in \mathcal{X}} S_u$$

$$m_O(\mathcal{X}) = \bigcup_{(l:S_m \vdash_{S_u} s \rightsquigarrow l') \in \mathcal{X}} \{m(S_m, S_m \cup S_u)\}$$

in that, for all $i = 0, \ldots, n$, $\mathcal{P}'_i = m_P(\mathcal{P}_i)$ and $\mathcal{O}'_i = m_O(\mathcal{O}_i)$. Here, $m(S, S')$ stands for the set S' where all sentences in $S \subseteq S'$ are marked, in the sense of Notation 1. As in Section 4, we assume that unless explicitly marked sentences are unmarked. We prove by induction (on the length n of the structured X-dispute derivation) that X-dd resulting from this construction is a X-dispute derivation:

Base case: step 0. By definition of structured X-dispute derivation and by construction, the initial tuple of X-dd is

$$\langle m_P(\{l_1 : \{\} \vdash_{\{\delta\}} \delta \rightsquigarrow \emptyset\}), m_O(\{\}), \mathcal{A} \cap \{\delta\}, \{\}, \{\}, \{\}\rangle$$

namely

$$\langle \{\delta\}, \{\}, \mathcal{A} \cap \{\delta\}, \{\}, \{\}, \{\}\rangle$$

which is the initial tuple of a X-dispute derivation.

Inductive hypothesis: step k, $0 \leqslant k < n$. Assume that $\langle m_P(\mathcal{P}_k), m_O(\mathcal{O}_k), D_k, C_k, \mathcal{F}_k\rangle$ in X-dd is obtained at step k, according to Definition 4.6.

Inductive step: step $k+1$. We prove that

$$\langle m_P(\mathcal{P}_{k+1}), m_O(\mathcal{O}_{k+1}), D_{k+1}, C_{k+1}, \mathcal{F}_{k+1}\rangle$$

is obtained at step $k+1$ in X-dd, by applying case x in Definition 4.6 if case x in Definition 6.3 has resulted in

$$\langle \mathcal{P}_{k+1}, \mathcal{O}_{k+1}, D_{k+1}, C_{k+1}, \mathcal{F}_{k+1}, Args_{k+1}, Att_{k+1}\rangle$$

in sX-dd, for suitable choices of selection function sel' and $member\mathcal{O}'$, and for the same choices of *turn*, $member\mathcal{F}$, *updt*, f_{DbyD}, f_{DbyC}, f_{CbyD} and f_{CbyC} as in sX-dd:

$x = 1$(i): If $member\mathcal{P}(\mathcal{P}_k) = (l : S_m \vdash_{S_u} s \rightsquigarrow l')$ and $sel(S_u) = \sigma$, by inductive hypothesis $\sigma \in \mathcal{P}'_k$. Let $sel'(\mathcal{P}_k) = \sigma$. It is easy to see that

$$m_O(\mathcal{O}_{k+1}) = m_O(\mathcal{O}_k) \cup \{\{\overline{\sigma}\}\}$$

$$m_P(\mathcal{P}_{k+1}) = m_P(\mathcal{P}_k) - \{\sigma\}$$

Thus, $\langle m_P(\mathcal{P}_{k+1}), m_O(\mathcal{O}_{k+1}), D_{k+1}, C_{k+1}, \mathcal{F}_{k+1}\rangle$ is a legitimate $k+1$-th tuple in X-dd according to Definition 4.6.

$x = 1$(ii): Similarly to case 1(i), let $sel'(\mathcal{P}_k) = \sigma$. Since $\mathcal{O}_{k+1} = \mathcal{O}_k$, trivially $m_O(\mathcal{O}_{k+1}) = m_O(\mathcal{O}_k)$. Moreover,

$$m_P(\mathcal{P}_{k+1}) = m_P(\mathcal{P}_k) \cup (S_u - \{\sigma\}) \cup f_{DbyD}(R, D_i)$$

Since $S_u \subseteq \mathcal{P}'_k = m_P(\mathcal{P}_k)$ by inductive hypothesis, we obtain

$$m_P(\mathcal{P}_k) \cup (S_u - \{\sigma\}) = m_P(\mathcal{P}_k) - \{\sigma\}$$

Thus,

$$m_P(\mathcal{P}_{k+1}) = m_P(\mathcal{P}_k) - \{\sigma\} \cup f_{DbyD}(R, D_i)$$

and $\langle m_P(\mathcal{P}_{k+1}), m_O(\mathcal{O}_{k+1}), D_{k+1}, C_{k+1}, \mathcal{F}_{k+1}\rangle$ is a legitimate $k+1$-th tuple in X-dd according to Definition 4.6.

$x = 2$(i)(a): If $member\mathcal{O}(\mathcal{O}_k) = (l : S_m \vdash_{S_u} s \rightsquigarrow l')$ and $sel(S_u) = \sigma$, by inductive hypothesis $m(Sm, S_m \cup S_u) \in \mathcal{O}'_k$ and $\sigma \in S_u$. Let $member\mathcal{O}(\mathcal{O}'_k) = m(S_m, S_m \cup S_u)$ and $sel'(S_u) = \sigma$. Since $\mathcal{P}_{k+1} = \mathcal{P}_k$, trivially $m_P(\mathcal{P}_{k+1}) = m_P(\mathcal{P}_k)$. Moreover,

$$m_O(\mathcal{O}_{k+1}) = m_O(\mathcal{O}_k) - \{m(S_m, S_m \cup S_u)\} \cup \{m(S_m \cup \{\sigma\}, S_m \cup S_u)\}$$

Thus, $\langle m_P(\mathcal{P}_{k+1}), m_O(\mathcal{O}_{k+1}), D_{k+1}, C_{k+1}, \mathcal{F}_{k+1}\rangle$ is a legitimate $k+1$-th tuple in X-dd according to Definition 4.6.

$x = 2$(i)(b): Similarly to case 2(i)(a), let $member\mathcal{O}(\mathcal{O}'_k) = m(S_m, S_m \cup S_u)$ and $sel'(S_u) = \sigma$. Since $\mathcal{P}_{k+1} = \mathcal{P}_k$, trivially $m_P(\mathcal{P}_{k+1}) = m_P(\mathcal{P}_k)$. Moreover, $m_O(\mathcal{O}_{k+1}) = m_O(\mathcal{O}_k) - \{m(S_m, S_m \cup S_u)\}$. Thus, $\langle m_P(\mathcal{P}_{k+1}), m_O(\mathcal{O}_{k+1}), D_{k+1}, C_{k+1}, \mathcal{F}_{k+1}\rangle$ is a legitimate $k+1$-th tuple in X-dd according to Definition 4.6.

$x = 2$(i)(c): Similarly to case 2(i)(a), let $member\mathcal{O}(\mathcal{O}'_k) = m(S_m, S_m \cup S_u)$ and $sel'(S_u) = \sigma$. Similarly to case 2(ii)(b), $m_O(\mathcal{O}_{k+1}) = m_O(\mathcal{O}_k) - \{m(S_m, S_m \cup S_u)\}$. Moreover, since in this case $\mathcal{P}_{k+1} = \mathcal{P}_k \cup \{l^* : \{\} \vdash_{\{\overline{\sigma}\}} \overline{\sigma} \rightsquigarrow l\}$, $m_P(\mathcal{P}_{k+1}) = m_P(\mathcal{P}_k) \cup \{\overline{\sigma}\}$. Thus, $\langle m_P(\mathcal{P}_{k+1}), m_O(\mathcal{O}_{k+1}), D_{k+1}, C_{k+1}, \mathcal{F}_{k+1}\rangle$ is a legitimate $k+1$-th tuple in X-dd according to Definition 4.6.

$x = 2$(ii): Similarly to case 2(i)(a), let $member\mathcal{O}(\mathcal{O}'_k) = m(S_m, S_m \cup S_u)$ and $sel'(S_u) = \sigma$. Since $\mathcal{P}_{k+1} = \mathcal{P}_k$, trivially $m_P(\mathcal{P}_{k+1}) = m_P(\mathcal{P}_k)$. It is easy to see, similarly to the earlier cases, that $m_O(\mathcal{O}_{k+1}) = m_O(\mathcal{O}_k) - \{m(S_m, S_m \cup S_u)\} \cup \{m(S_m, (S_m - \{\sigma\}) \cup R) | \sigma \leftarrow R \text{ and not } f_{CbyC}(R, C_i)\}$.
Thus, $\langle m_P(\mathcal{P}_{k+1}), m_O(\mathcal{O}_{k+1}), D_{k+1}, C_{k+1}, \mathcal{F}_{k+1}\rangle$ is a legitimate $k+1$-th tuple in X-dd according to Definition 4.6.

$x = 3$: Since this step only modifies the $\mathcal{F}$ component, trivially $\langle m_P(\mathcal{P}_{k+1}), m_O(\mathcal{O}_{k+1}), D_{k+1}, C_{k+1}, \mathcal{F}_{k+1}\rangle$ is a legitimate $k+1$-th tuple in X-dd according to Definition 4.6.

Trivially, sX-dd and X-dd compute the same support D_n.

From a X-dispute derivation to a structured X-dispute derivation

Finally, we sketch how to construct a structured X-dispute derivation sX-dd

$$\langle \mathcal{P}'_0, \mathcal{O}'_0, D_0, C_0, \mathcal{F}_0, Args_0, Att_0\rangle, \quad \ldots, \quad \langle \mathcal{P}'_n, \mathcal{O}'_n, D_n, C_n, \mathcal{F}_n, Args_n, Att_n\rangle$$

from a given X-dispute derivation X-dd:

$$\langle \mathcal{P}_0, \mathcal{O}_0, D_0, C_0, \mathcal{F}_0\rangle, \quad \ldots, \quad \langle \mathcal{P}_n, \mathcal{O}_n, D_n, C_n, \mathcal{F}_n\rangle$$

This construction relies upon the inverses of the mappings m_P and m_O defined earlier, in that $m_P(\mathcal{P}'_i) = \mathcal{P}_i$ and $m_O(\mathcal{O}'_i) = \mathcal{O}_i$ for all $i = 0, \ldots, n$. The construction again mirrors all choices of parameters as before, giving new sel', $member\mathcal{O}'$ and $member\mathcal{P}$ in sX-dd and the same other choices in sX-dd as in X-dd. The construction is inductive. We will focus on the construction of $Args_i$ and Att_i in sX-dd.

Base case: step 0. $Args_0, Att_0$ can be trivially constructed so as to match Definition 6.3.

Inductive hypothesis: step k, $0 \leqslant k < n$. Assume that $\langle \mathcal{P}'_k, \mathcal{O}'_k, D_k, C_k, \mathcal{F}_k, Args_k, Att_k\rangle$ in sX-dd is constructed at step k in such a way that $m_P(\mathcal{P}'_k) = \mathcal{P}_k$ and $m_O(\mathcal{O}'_k) = \mathcal{O}_k$ and so as to satisfy Definition 6.3.

Inductive step: step $k+1$. The only cases where the *Args* and *Att* components may be updated are $x = 1$(i), 1(ii), 2(i)(b) or 2(ii).

$x = 1$(i): By inductive hypothesis, if $\mathcal{P}_k - \{\sigma\} = \{\}$ then $\mathcal{P}'_{k+1} = \mathcal{P}'_k - \{\pi\}$ for $\pi = (l : S_m \vdash_{S_u} s \rightsquigarrow l')$ such that $member\mathcal{P}'(\mathcal{P}'_k) = \pi$ and $sel'(S_u) = \sigma$. In this case, $Args_{k+1}$ and Att_{k+1} are modified so as to satisfy Definition 6.3. Moreover, even if $\mathcal{P}_k - \{\sigma\} \neq \{\}$, it may be that $S_u = \{\sigma\}$ in the selected π (by using the sel' mirroring the original sel). In this case, $Args_{k+1}$ and Att_{k+1} are modified so as to satisfy Definition 6.3.

$x = 1$(ii): By inductive hypothesis, if $sel(\mathcal{P}_k) = \sigma$, there exists $\pi = (l : S_m \vdash_{S_u} s \rightsquigarrow l')$ such that $member\mathcal{P}'(\mathcal{P}'_k) = \pi$ and $sel'(S_u) = \sigma$. Then, depending on the outcome of $f_{DbyD}(R, D_k)$ for the $\sigma \leftarrow R$ chosen in X-dd, $Args_{k+1}$ and Att_{k+1} are modified so as to satisfy Definition 6.3.

$x = 2$(i)(b): Similar to case 1(ii).

$x = 2$(ii): Similar to case 1(ii).

B.2. Preliminaries for the proofs of Theorems 7.2 and 7.3

First, we give some preliminary notions/results, for the dialectical structure $(Args, Att)$ computed by a structured X-dispute derivation for a sentence δ w.r.t. any choices of parameters amongst GB-, AB- and IB-choices, as defined in Section 6, and for $(Args^a, Att^a) = Actual(Args, Att)$ and $\mathcal{T}^*(Args, Att)$ and $\mathcal{F}(Args^a, Att^a)$, as defined in Section 7. We will make use of the fact that all choices of parameters are canonical (as is the case for GB-, AB- and IB-choices).

Let us refer to all potential arguments introduced in *Args* at steps 1(i) or 1(ii) (steps 2(i)(b) or 2(i)(c) or 2(ii)) in the definition of structured X-dispute derivations and the actual arguments in $Args^a$ obtained from them as $\mathcal{P}$-*arguments* ($\mathcal{O}$-*arguments*, respectively). It is easy to see that the argument in *Args* labelled l_1, where $l_1 : \{\} \vdash_{\{\delta\}} \delta \rightsquigarrow \emptyset \in \mathcal{P}_0$, is necessarily a $\mathcal{P}$-argument. Also, trivially all $\mathcal{P}$-arguments in *Args* are actual arguments.

Lemma B.1. *Let N be any node in any of the trees in* $\mathcal{F}(Args^a, Att^a)$. *Then N is odd-level* (*even-level*) *iff N holds a* $\mathcal{P}$-*argument* ($\mathcal{O}$-*argument, respectively*).

Proof of Lemma B.1. Trivially, since $\mathcal{P}$-arguments all have an empty unmarked support, whereas $\mathcal{O}$-arguments may have a non-empty unmarked support: All $\mathcal{P}$-arguments in *Args* are actual arguments and all potential arguments in *Args* are $\mathcal{O}$-arguments. Further, each $l' \rightsquigarrow l'' \in Att$ is obtained from some $l' : A \vdash_S \sigma \rightsquigarrow l''$ in either some $\mathcal{P}_i$ or some $\mathcal{O}_j$. By definition of structured X-dispute derivation, for each $l' : A \vdash_S \sigma \rightsquigarrow l''$ in some $\mathcal{P}_i$ ($\mathcal{O}_j$), there exists a $l'' : A' \vdash_{S'} \sigma' \rightsquigarrow l^*$ in some $\mathcal{O}_j$ ($\mathcal{P}_i$ respectively), Thus, by construction of $\mathcal{T}^*(Args, Att)$, children of $\mathcal{P}$-arguments in $\mathcal{T}^*(Args, Att)$ are $\mathcal{O}$-arguments, and children of $\mathcal{O}$-arguments in $\mathcal{T}^*(Args, Att)$ are $\mathcal{P}$-arguments. Thus $\mathcal{P}$-arguments ($\mathcal{O}$-arguments) in *Args* label odd-level (even-level, respectively) nodes in $\mathcal{T}^*(Args, Att)$. Then, it follows directly that all odd-level nodes in $\mathcal{T}^*(Args, Att)$ hold actual arguments, and all potential arguments in $\mathcal{T}^*(Args, Att)$ are held at even-level nodes. By definition of dialectical forest, potential arguments are eliminated (and possibly replaced by actual arguments) in a dialectical forest. As a consequence the lemma holds. □

Lemma B.1 will be used to prove that trees in (expanded) forests fulfil conditions 1 and 2 in the definition of the dispute trees of [13,14]. The following Lemma B.2 will be used to prove that trees in (expanded) forests fulfil condition 3 of dispute trees.

Lemma B.2. *For every* $\mathcal{P}$-*argument* $\mathtt{b}$ *in* (*some tree in*) $\mathcal{F}(Args^a, Att^a)$, *for every argument* $\mathtt{c}$ *that attacks* $\mathtt{b}$ (*in the underlying ABA framework*), *there is an even-level node N in* (*some tree in*) $\mathcal{F}(Args^a, Att^a)$ *such that N holds* $\mathtt{c}$.

Proof of Lemma B.2. Assume $\mathtt{b}$ is $l : A \vdash_{\{\}} \sigma$. Since $\mathtt{b}$ is a $\mathcal{P}$-argument, there exists a step i in the given structured X-dispute derivation such that $\mathtt{b} \in Args_{i+1} - Args_i$, $l : A' \vdash_{S'} \sigma \rightsquigarrow l' \in \mathcal{P}_i$, $l : \{\} \vdash_{\{\sigma\}} \sigma \rightsquigarrow l' \in \mathcal{P}_{i_0}$ for some $i_0 \leqslant i$ and, for each $i_0 \leqslant j \leqslant i$, $l : A'_j \vdash_{S'_j} \sigma \rightsquigarrow l' \in \mathcal{P}_j$ for some A'_j, S'_j. Consider an argument $\mathtt{c}$ attacking $\mathtt{b}$ (if there is no argument attacking $\mathtt{b}$ then the lemma is trivially true). Then, $\mathtt{c}$ is a proof for some σ' such that $\sigma' = \overline{\beta}$ for some $\beta \in A$. Necessarily, $\beta \in A'_j \cup S'_j$ for some $i_0 \leqslant j \leqslant i$, and

1. either $\beta \in S'_j$ and $sel(S'_j) = \beta$, for some $i_0 \leqslant j \leqslant i$;
   then, by step 1(i), $l^* : \{\} \vdash_{\{\sigma'\}} \sigma' \rightsquigarrow l \in \mathcal{O}_{j+1}$; trivially, since $\mathcal{O}_n = \{\}$ and by definition of step 2(ii) in structured X-dispute derivations, there exists some $l^*(R_1) \ldots (R_k) : S \vdash_Z \sigma' \in Args$, for $0 \leqslant k \leqslant n$ and each $R_l \subseteq \mathcal{L}$, such that $\mathtt{b}$ is a proof for σ' supported by (some) W and expanding $S \vdash_Z \sigma'$; thus, necessarily $\mathtt{c} \in Args^a$ and $\mathtt{c}$ is a node in some tree in $\mathcal{F}(Args^a, Att^a)$;
2. or $\beta \notin S'_j$ for any $i_0 \leqslant j \leqslant i$ and $\beta \in (R - f_{DbyD}(R, D_k))$ for some $i_0 \leqslant k \leqslant i$, and some $R \subseteq \mathcal{L}$;
   this cannot be the case for GB-choices of parameters (as these force $f_{DbyD}(R, D_k) = R$); for AB- or IB-choices, where $f_{DbyD}(R, D_k) = R - D_k$, this implies that $\beta \in D_k$ and there exists $l^\dagger : A^\dagger \vdash_{S^\dagger} \sigma^\dagger \rightsquigarrow l^{\dagger\dagger} \in \mathcal{P}_{k_j}$ (with $k_j < k$) with $\beta \in S^\dagger$ and $sel(S^\dagger) = \beta$; similarly to the previous case, this implies that $\mathtt{c}$ is a node; thus, necessarily $\mathtt{c} \in Args^a$ and $\mathtt{c}$ is a node in some tree in $\mathcal{F}(Args^a, Att^a)$.

This holds for any such $\mathtt{c}$, and thus the lemma is proved. □

Note that, in general, $\mathtt{b}$ and $\mathtt{c}$ may be held at nodes in different trees in the forest, and that, even when $\mathtt{b}$ and $\mathtt{c}$ are in the same tree, $\mathtt{c}$ may be a child of $\mathtt{b}$ or not. However, for GB-choices of parameters, $\mathtt{c}$ will always be a child of $\mathtt{b}$ in the same tree:

Lemma B.3. *For every* $\mathcal{P}$-*argument* $\mathtt{b}$ *in some tree* $\mathcal{T}$ *in* $\mathcal{F}(Args^a, Att^a)$ *w.r.t. GB-choices of parameters, for every argument* $\mathtt{c}$ *that attacks* $\mathtt{b}$ (*in the underlying ABA framework*), $\mathtt{c}$ *labels a child of* $\mathtt{b}$ *in* $\mathcal{T}$.

Proof of Lemma B.3. For GB-choices, only case 1 in the proof of Lemma B.2 can arise. Since in this case $l^* : \{\} \vdash_{\{\sigma'\}} \sigma' \rightsquigarrow l \in \mathcal{O}_{j+1}$, necessarily $l^* \rightsquigarrow l \in Att$ and thus $\mathtt{c}$ (labelled l^*) is a child of $\mathtt{b}$ (labelled by l) by definition of forest. □

The following Lemma B.4 will be used to prove that trees in (expanded) forests fulfil condition 4 in the definition of the dispute trees of [13,14].

Lemma B.4. *For every $\mathcal{O}$-argument $\mathtt{b}$ somewhere in (some tree in) $\mathcal{F}(Args^a, Att^a)$, there exists an odd-level node N in (some tree in) $\mathcal{F}(Args^a, Att^a)$, such that N holds an argument $\mathtt{c}$ that attacks $\mathtt{b}$ (in the underlying ABA framework).*

Proof of Lemma B.4. Assume $\mathtt{b}$ is $l : A \vdash_{\{\}} \sigma$ and let $l' : A' \vdash_{S'} \sigma \in Args$ be the potential argument from which $\mathtt{b}$ is obtained in the transition from $(Args, Att)$ to $Actual(Args, Att)$. Since $\mathtt{b}$ is an $\mathcal{O}$-argument, $l' : A' \vdash_{S'} \sigma$ is an even-level node in $\mathcal{T}^*(Args, Att)$. Also, there exists a step i in the given structured X-dispute derivation such that $l' : A' \vdash_{S'} \sigma \in Args_{i+1} - Args_i$ and some $\alpha \in A'$ such that

1. either step i is of the 2(i)(c) kind and $l' : (A' - \{\alpha\}) \vdash_{(S' \cup \{\alpha\})} \sigma \rightsquigarrow l'' \in \mathcal{O}_i$, for some label l''; thus, by definition of step 2(i)(c), some $l^* : \{\} \vdash_{\{\overline{\alpha}\}} \overline{\alpha} \rightsquigarrow l' \in \mathcal{P}_{i+1}$ and, since $\mathcal{P}_n = \{\}$ and by definition of structured X-dispute derivation, some $l^* : \Sigma \vdash_{\{\}} \overline{\alpha} \in Args$;
2. or step i is of the 2(i)(b) kind and $f_{CbyC}(\{\alpha\}, C_i)$ holds; thus, since f_{CbyC} is canonical, necessarily $\alpha \in C_i$; thus, by definition of structured X-dispute derivation, there exists a step $j < i$ of the 2(i)(c) kind, in the given structured X-dispute derivation such that $l^\dagger : A^\dagger \vdash_{S^\dagger} \sigma^\dagger \in Args_{j+1} - Args_j$ and with $\alpha \in A^\dagger$ and, similarly to the previous case 1, some $l^* : \Sigma \vdash_{\{\}} \overline{\alpha} \in Args$.

In either case, there exists an argument $\mathtt{c}$ (this is $l^* : \Sigma \vdash_{\{\}} \overline{\alpha}$) attacking $\mathtt{b}$ in the forest. □

Note that, in general, $\mathtt{b}$ and $\mathtt{c}$ may be held at nodes in different trees in the forest, and that, even when $\mathtt{b}$ and $\mathtt{c}$ are in the same tree, $\mathtt{c}$ may be a child of $\mathtt{b}$ or not. However, for GB-choices of parameters, $\mathtt{c}$ will always be a child of $\mathtt{b}$ in the same tree:

Lemma B.5. *For every $\mathcal{O}$-argument $\mathtt{b}$ in some tree $\mathcal{T}$ in $\mathcal{F}(Args^a, Att^a)$ w.r.t. GB-choices of parameters, there exists a child of $\mathtt{b}$ in $\mathcal{T}$ labelled by an argument $\mathtt{c}$ that attacks $\mathtt{b}$ (in the underlying ABA framework).*

Proof of Lemma B.5. For GB-choices, only case 1 in the proof of Lemma B.4 can arise. The lemma thus follows as in the case of Lemma B.3. □

Finally, the following lemma will be used to prove that trees in expanded forests are admissible:

Lemma B.6. *There exists no argument in $Args^a$ that is both a $\mathcal{P}$-argument and an $\mathcal{O}$-argument.*

Proof of Lemma B.6. By contradiction, assume there is $A \vdash_{\{\}} \sigma \in Args^a$ that is both a $\mathcal{P}$-argument and an $\mathcal{O}$-argument. Since all $\mathcal{P}$-arguments in $Args$ are necessarily actual, $A \vdash_{\{\}} \sigma$ is a $\mathcal{P}$-argument in $Args$. Let i be the step in the given structured X-dispute derivation such that $A \vdash_{\{\}} \sigma \in Args_{i+1} - Args_i$. Trivially, $A \subseteq D_{i+1}$ (*). Also, $A \cap C_i = \{\}$ (**), since, if $i = 1$ then $C_1 = \{\}$, and, for $i > 1$ since f_{DbyC} is canonical.

Since $A \vdash_{\{\}} \sigma$ is also an $\mathcal{O}$-argument in $Args^a$, either (1) $A \vdash_{\{\}} \sigma \in Args$ or (2) there exists $A' \vdash_{S'} \sigma \in Args$ and a proof for σ supported by A and expanding $A' \vdash_{S'} \sigma$. Let j be the step in the given structured X-dispute derivation such that (1) either $A \vdash_{\{\}} \sigma \in Args_{j+1} - Args_j$ or (2) $A' \vdash_{S'} \sigma \in Args_{j+1} - Args_j$. Then, there exists $\alpha \in A$ (case 1) or $\alpha \in A' \subseteq A$ (case 2) such that $f_{CbyD}(\alpha, D_j) = true$. Either $j > i$ or $j < i$:

if $j > i$ then, since $A \subseteq D_{i+1} \subseteq D_j$ (by (*)), necessarily $\alpha \in D_j$ and thus f_{CbyD} is not canonical: contradiction;
if $j < i$ then $\alpha \in C_{j+1} \subseteq C_i$, and then (**) does not hold: contradiction. □

B.3. Proof of Theorem 7.2

(A) Trivially, every tree in $\mathcal{F}(Args^a, Att^a)$ is finite, so we only need to prove that every such tree is a dispute tree, as reviewed in Section 2.
   1. By Lemma B.1, each node has either proponent (if odd-level) or opponent (if even-level) status, but not both.
   2. By Lemma B.1, adopting the status assignment in 1, the root of every tree in $\mathcal{F}(Args^a, Att^a)$ is a proponent node.
   3. Directly from Lemma B.3.
   4. Directly from Lemma B.5.
   5. Trivial, by definition of structured X-dispute derivation and of forest.

(B) It is easy to see that there exists an argument for δ in $Args$, and this is labelled by some l such that $l \rightsquigarrow \emptyset \in Att$. Thus, this argument is the root of $\mathcal{T}^*(Args, Att)$ and of some tree $\mathcal{T}$ in $\mathcal{F}(Args^a, Att^a)$. By part (A), this $\mathcal{T}$ is necessarily a grounded dispute tree. □

B.4. Proof of Lemma 7.1

This lemma trivially follows from Lemmas B.2 and B.4, by definition of attack.

B.5. Proof of Theorem 7.3

We first prove the theorem for *AB-choices of parameters and admissible dispute trees*.

(A) Every tree in the expanded forest of $\mathcal{F}^p(Args^a, Att^a)$ is a dispute tree:
1. By Lemma B.1, each node in the forest $\mathcal{F}^p(Args^a, Att^a)$ has either proponent (if odd-level) or opponent (if even-level) status, but not both. By definition, all newly added nodes in the expanded forest are nodes in the original forest, and they have the same status as in the original trees. Thus, each node in the expanded forest of $\mathcal{F}^p(Args^a, Att^a)$ has either proponent (if odd-level) or opponent (if even-level) status, but not both.
2. By adopting the status assignment in 1, the root of every tree in the expanded forest of $\mathcal{F}(Args^a, Att^a)$ is a proponent node.
3. Directly from Lemma B.2 and by construction of expanded forest.
4. Directly from Lemma B.4 and by construction of expanded forest.
5. Trivial, by definition of structured X-dispute derivation and of expanded forest.

Finally, by Lemma B.6, every tree in the expanded forest of $\mathcal{F}^p(Args^a, Att^a)$ is admissible.

(B) It is easy to see that there exists an argument for δ in *Args*, and this is labelled by some l such that $l \rightsquigarrow \emptyset \in Att$. Thus, this argument is the root of $\mathcal{T}^*(Args, Att)$ and of some tree $\mathcal{T}$ in the expanded forest of $\mathcal{F}(Args^a, Att^a)$. By part (A), this $\mathcal{T}$ is an admissible dispute tree. □

Let us now consider the case of *IB-choices of parameters and ideal dispute trees*. By virtue of the earlier proof for the AB-choices of parameters and admissible dispute trees, we only need to prove that for no even-level node N in any of the trees in the expanded forest of $\mathcal{F}(Args^a, Att^a)$ there exists an admissible dispute tree for the argument held at N. By definition of structured X-dispute derivation and of IB-choices of parameters, the support[11] of every potential argument added to $\mathcal{O}_i$ is also added to $\mathcal{F}_i$ (at steps 2(i)(b) or 2(i)(c) or 2(ii)), and, for each such support S, $Fail(S)$ holds (due to step 3). By Definition 3.1 of *Fail*, this means that, for each such S, there exists no admissible $E \subseteq \mathcal{A}$ such that, for each $\sigma \in S$, there exists an argument for σ supported by some subset of E. By contradiction, assume that there is even-level node N and an admissible dispute tree for the argument $A \vdash_{\{\}} \sigma'$ held at N. By Theorem 3.1(i) in [14], there exists an admissible $E \subseteq \mathcal{A}$ such that $A \subseteq E$. Since $A \vdash_{\{\}} \sigma'$ is a proof for σ' supported by A and expanding some (opponent argument) $A' \vdash_{S'} \sigma' \in Args$ (with $A' \subseteq A$), then E is an admissible set of assumptions such that arguments for all sentences in $A' \cup S'$ are supported by arguments with support a subset of E. Thus, $Fail(A' \cup S')$ cannot possibly hold, but $A' \cup S' \in \mathcal{F}_i$ for some i: contradiction. □

Appendix C. Proofs for Section 8

C.1. Proof of Theorem 8.1

For AB-choices and admissible semantics this theorem directly follows from Proposition 5.2 in Section 5 and Theorem 4.4 in [14]. For IB-choices and ideal semantics it directly follows from Proposition 5.3 in Section 5 and Theorem 4.6 in [14]. Thus, we only need to prove the theorem for GB-choices (namely f_{DbyD}, f_{DbyC}, f_{CbyD}, f_{CbyC} and *updt* as in Definition 5.1) and grounded semantics.

If $\mathtt{a} \in \mathbb{A}$ and $\mathbb{A}$ is grounded then there exists a grounded (i.e. finite) dispute tree $\mathcal{T}$ for $\mathtt{a}$ with argument defence set $\mathbb{A}'$ such that $\mathbb{A}' \subseteq \mathbb{A}$ (see Section 2) and thus $Asm(\mathbb{A}') \subseteq Asm(\mathbb{A})$. We construct a X-dispute derivation of support $\Delta \subseteq Asm(\mathbb{A}')$ from this tree and obtain specific (canonical) choices for *sel*, *member*$\mathcal{O}$, *member*$\mathcal{F}$, and *turn* in the process.

Let $N_1, \ldots, N_k$, $k \geqslant 1$, be a left-most depth-first order on the nodes of $\mathcal{T}$. Each node N_i is an argument for some sentence σ_i, namely a tree $\mathcal{T}_{N_i}$ with root σ_i. Trivially, the root of $\mathcal{T}_{N_1}$ is σ and $\mathcal{T}_{N_1}$ is a. Following [13], each tree $\mathcal{T}_{N_i}$ can be equivalently seen as a backward deduction, namely a sequence of sets of sentences $S^i_1, \ldots, S^i_{k_i}$ ($k_i \geqslant 1$), corresponding to frontiers obtained during the tree construction, and omitting any occurrences of τ. Below, we adopt the following conventions:

- $support(N_i)$ is the support of the argument at node N_i,
- $culprit(N_i) = \begin{cases} \{\chi\} & \text{if } N_i \text{ is an opponent node, } \chi \in support(N_i), \overline{\chi} = \sigma_{i+1} \\ \{\} & \text{if } N_i \text{ is a proponent node.} \end{cases}$

Let N_i be a proponent node, for $1 \leqslant i \leqslant k$, let $c_i \geqslant 0$ be the cardinality of $support(N_i)$, and $\alpha^i_1, \ldots, \alpha^i_{c_i}$ some order over the assumptions in $support(N_i)$. Let $seq(N_i)$ be

$$\langle \mathcal{P}^i_1, \mathcal{O}^i_1, D^i_1, C^i_1, \mathcal{F}^i_1 \rangle, \quad \ldots, \quad \langle \mathcal{P}^i_{k_i}, \mathcal{O}^i_{k_i}, D^i_{k_i}, C^i_{k_i}, \mathcal{F}^i_{k_i} \rangle,$$
$$\langle \mathcal{P}^i_{k_i+1}, \mathcal{O}^i_{k_i+1}, D^i_{k_i+1}, C^i_{k_i+1}, \mathcal{F}^i_{k_i+1} \rangle, \quad \ldots, \quad \langle \mathcal{P}^i_{k_i+c_i}, \mathcal{O}^i_{k_i+c_i}, D^i_{k_i+c_i}, C^i_{k_i+c_i}, \mathcal{F}^i_{k_i+c_i} \rangle$$

where

[11] The support of a potential argument $X \vdash_Y x$ is $X \cup Y$.

- $\mathcal{F}^i_j = \{\}$, for $j = 1, \ldots, k_i, k_i + 1, \ldots, k_i + c_i$;
- $\mathcal{P}^i_j = S^i_j$, for $j = 1, \ldots, k_i$,
  $\mathcal{P}^i_{k_i+j+1} = \mathcal{P}_{k_i+j} - \{\alpha^i_{j+1}\}$, for $j = 0, \ldots, c_i - 1$;
- $\mathcal{O}^i_j = \{\}$, for $j = 1, \ldots, k_i$,
  $\mathcal{O}^i_{k_i+j+1} = \mathcal{O}_{k_i+j} \cup \{\{\overline{\alpha^i_{j+1}}\}\}$, for $j = 0, \ldots, c_i - 1$;
- $C^i_j = C^i_1$, for $j = 2, \ldots, k_i + c_i$,
  $C^i_1 = \{\}$, for $i = 1$,
  $C^i_1 = \bigcup_{1<o<i} culprit(N_o)$, for $i > 1$;
- $D^i_{j+1} = D^i_j \cup (\mathcal{P}^i_{j+1} \cap \mathcal{A})$, for $j = 1, \ldots, k_i + c_i - 1$,
  $D^i_1 = \{\sigma\} \cap \mathcal{A}$, for $i = 1$,
  $D^i_1 = \bigcup_{1<p<i, N_p}$ is a proponent node $support(N_p) \cup (\mathcal{P}^i_1 \cap \mathcal{A})$, for $i > 1$.

Trivially, $\mathcal{P}^i_{k_i+c_i} = \{\}$. Now, let N_i be an opponent node, for $2 \leqslant i \leqslant k$, and let $\chi^i = culprit(N_i)$. Let $seq(N_i)$ be
$\langle \mathcal{P}^i_1, \mathcal{O}^i_1, D^i_1, C^i_1, \mathcal{F}^i_1 \rangle, \ldots, \langle \mathcal{P}^i_{k_i}, \mathcal{O}^i_{k_i}, D^i_{k_i}, C^i_{k_i}, \mathcal{F}^i_{k_i} \rangle, \langle \mathcal{P}^i_{k_i+1}, \mathcal{O}^i_{k_i+1}, D^i_{k_i+1}, C^i_{k_i+1}, \mathcal{F}^i_{k_i+1} \rangle$,
where

- $\mathcal{F}^i_j = \{\}$, for $j = 1, \ldots, k_i + 1$;
- $\mathcal{P}^i_j = \{\}$, for $j = 1, \ldots, k_i$,
  $\mathcal{P}^i_{k_i+1} = \{\overline{\chi^i}\}$;
- $\mathcal{O}^i_1 = \bigcup_{\alpha \in support(N_{i-1})} \{\{\overline{\alpha}\}\}$,[12]
  $\mathcal{O}^i_j = \mathcal{O}^i_{j-1} - \{S^i_{j-1}\} \cup \{S^i_j\}$, for $j = 2, \ldots, k_i$,
  $\mathcal{O}^i_{k_i+1} = \mathcal{O}^i_{k_i} - \{S^i_{k_i}\}$;
- $C^i_j = C^i_1$, for $j = 2, \ldots, k_i$,
  $C^i_1 = \bigcup_{1<o<i} culprit(N_o)$,
  $C^i_{k_i+1} = C^i_{k_i} \cup \{\chi^i\}$;
- $D^i_{j+1} = D^i_j$, for $j = 1, \ldots, k_i$,
  $D^i_1 = \bigcup_{1<p<i, N_p \text{ is a proponent node}} support(N_p)$, for $i > 1$,
  $D^i_{k_i+1} = D^i_{k_i} \cup (\{\overline{\chi^i}\} \cap \mathcal{A})$.

It is easy to see that $\mathcal{O}^i_{k_i+1}$ may be non-empty. Indeed, there may be some assumptions α in $support(N_{i-1})$ that cannot be attacked. Elements in $\mathcal{O}^i_{k_i+1}$ are of the form $\{\overline{\alpha}\}$. Since we are dealing with p-acyclic frameworks, the search for arguments for $\overline{\alpha}$ fails finitely. Thus, trivially, $seq(N_i)$ can be extended by means of additional tuples corresponding to frontiers of proofs for the contraries $\overline{\alpha}$ of these assumptions, supported by sets of assumptions and non-assumptions. Only the $\mathcal{O}$ elements of these tuples will change, and all other elements will be as in the $k_i + 1$-th tuple. We will refer to this extension of $seq(N_i)$ as $e\_seq(N_i)$. The last tuple of $e\_seq(N_i)$ will have an empty $\mathcal{O}$ component.

We will use $e\_seq(N_i)$ to stand for $seq(N_i)$ when N_i is a proponent node.

It is easy to see that the last tuple in $e\_seq(N_i)$ is identical to the first tuple in $e\_seq(N_{i+1})$, for $i = 1, \ldots, k-1$. Let $e\_seq^-(N_{i+1})$ be $e\_seq(N_{i+1})$ without the first tuple, for $i = 1, \ldots, k-1$. Let SEQ be $seq(N_1), e\_seq^-(N_2), \ldots, e\_seq^-(N_k)$. Trivially, SEQ is a X-dispute derivation, for GB-choices of parameters, and appropriate choices of sel, $member\mathcal{O}$, $member\mathcal{F}$, and $turn$ (in particular, sel is patient). Moreover, it is easy to see that the support Δ computed by this X-dispute derivation is exactly $Asm(\mathbb{A}')$. □

C.2. Proof of Theorem 8.2

We prove the theorem by adapting and expanding the construction in the proof of Theorem C.1 to include *Args* and *Att* components, and modify the $\mathcal{P}$ and $\mathcal{O}$ components to include labelled potential arguments with potential attacks, as shown below. In doing so, we assume that *label* is generating labels $l_0, l^1_1, \ldots, l^{k_1}_1, \ldots, l^1_k, \ldots, l^{k_k}_k$ for potential arguments so that $l_0 = \emptyset$, $l^{k_j}_i = l_i$, for $i \geqslant 1$, and the argument held at node N_i is labelled l_i. We also indicate with $label(i, \sigma)$ a label, amongst $l_1, \ldots, l_k$, for a node, child of node N_i, holding an argument for σ (note that there may be several such nodes, and we basically assume some given order over them).

[12] Note that N_{i-1} is necessarily the parent of N_i, and is a proponent node.

For N_i a proponent node, $1 \leqslant i \leqslant k$, $seq(N_i)$ becomes

$$\langle \mathcal{P}^i_1, \mathcal{O}^i_1, D^i_1, C^i_1, \mathcal{F}^i_1, Args^i_1, Att^i_1\rangle, \ldots,$$
$$\langle \mathcal{P}^i_{k_i}, \mathcal{O}^i_{k_i}, D^i_{k_i}, C^i_{k_i}, \mathcal{F}^i_{k_i}, Args^i_{k_i}, Att^i_{k_i}\rangle,$$
$$\langle \mathcal{P}^i_{k_i+1}, \mathcal{O}^i_{k_i+1}, D^i_{k_i+1}, C^i_{k_i+1}, \mathcal{F}^i_{k_i+1}, Args^i_{k_i+1}, Att^i_{k_i+1}\rangle, \ldots,$$
$$\langle \mathcal{P}^i_{k_i+c_i}, \mathcal{O}^i_{k_i+c_i}, D^i_{k_i+c_i}, C^i_{k_i+c_i}, \mathcal{F}^i_{k_i+c_i}, Args^i_{k_i+c_i}, Att^i_{k_i+c_i}\rangle$$

where the D, C and $\mathcal{F}$ components are as before and

- $\mathcal{P}^i_j = \{l^j_i : \{\} \vdash_{S^i_j} \sigma\ i \rightsquigarrow l_{i-1}\}$, for $j = 1, \ldots, k_i$,
  $\mathcal{P}^i_{k_i+j+1} = \mathcal{P}_{k_i+j} - \{l_i : \{\alpha^i_1, \ldots, \alpha^i_j\} \vdash_{Support(N_i)-\{\alpha^i_1,\ldots,\alpha^i_j\}} \sigma_i \rightsquigarrow l_{i-1}\}$
  $\cup \{l_i : \{\alpha^i_1, \ldots, \alpha^i_j, \alpha^i_{j+1}\} \vdash_{Support(N_i)-\{\alpha^i_1,\ldots,\alpha^i_j,\alpha^i_{j+1}\}} \sigma_i \rightsquigarrow l_{i-1}\}$, for $j = 0, \ldots, c_i - 1$;
- $\mathcal{O}^i_j = \{\}$, or $j = 1, \ldots, k_i$,
  $\mathcal{O}^i_{k_i+j+1} = \mathcal{O}_{k_i+j} \cup \{label(i, \overline{\alpha^i_{j+1}})^1 : \{\} \vdash_{\overline{\{\alpha^i_{j+1}\}}} \overline{\alpha^i_{j+1}} \rightsquigarrow l_i\}$, for $j = 0, \ldots, c_i - 1$;
- $Args^i_j = \{l_x : Support(N_x) \vdash_{\{\}} \sigma_x | x = 1, \ldots, i-1\}$, for $j = 1, \ldots, k_i, k_i+1, \ldots, k_i+c_i-1$,
  $Args^i_{k_i+c_i} = \{l_x : Support(N_x) \vdash_{\{\}} \sigma_x | x = 1, \ldots, i\}$;
- $Att^i_j = \{l_1 \rightsquigarrow \emptyset\} \cup \{l_x \rightsquigarrow l_y | l_x : \_ \vdash_\_ \sigma_x, l_y : \_ \vdash_\_ \sigma_y \in Args^i_j$, and N_x is a child of N_y in $\mathcal{T}\}$, for $j = 1, \ldots, k_i, k_i+1, \ldots, k_i+c_i$.

Trivially, $\mathcal{P}^i_{k_i+c_i} = \{\}$.

For N_i an opponent node, for $2 \leqslant i \leqslant k$, and $\chi^i = culprit(N_i)$. $seq(N_i)$ becomes

$$\langle \mathcal{P}^i_1, \mathcal{O}^i_1, D^i_1, C^i_1, \mathcal{F}^i_1, Args^i_1, Att^i_1\rangle, \ldots,$$
$$\langle \mathcal{P}^i_{k_i}, \mathcal{O}^i_{k_i}, D^i_{k_i}, C^i_{k_i}, \mathcal{F}^i_{k_i}, Args^i_{k_i}, Att^i_{k_i}\rangle,$$
$$\langle \mathcal{P}^i_{k_i+1}, \mathcal{O}^i_{k_i+1}, D^i_{k_i+1}, C^i_{k_i+1}, \mathcal{F}^i_{k_i+1}, Args^i_{k_i+1}, Att^i_{k_i+1}\rangle$$

where the D, C and $\mathcal{F}$ components are as before and

- $\mathcal{P}^i_j = \{\}$, for $j = 1, \ldots, k_i$,
  $\mathcal{P}^i_{k_i+1} = \{label(i, \overline{\chi^i})^1 : \{\} \vdash_{\overline{\{\chi^i\}}} \overline{\chi^i} \rightsquigarrow l_i\}$;
- $\mathcal{O}^i_1 = \bigcup_{\alpha \in support(N_{i-1})} \{label(i-1, \overline{\alpha})^1 : \{\} \vdash_{\{\overline{\alpha}\}} \overline{\alpha} \rightsquigarrow l_{i-1}\}$,[13]
  $\mathcal{O}^i_j = \mathcal{O}^i_{j-1} - \{l^{j-1}_i : \{\} \vdash_{S^i_{j-1}} \sigma_i \rightsquigarrow l_{i-1}\}$
  $\cup \{\{l^j_i : \{\} \vdash_{S^i_j} \sigma_i \rightsquigarrow l_{i-1}\}$, for $j = 2, \ldots, k_i$,
  $\mathcal{O}^i_{k_i+1} = \mathcal{O}^i_{k_i} - \{l^{k_i}_i : \{\} \vdash_{S^i_{k_i}} \sigma_i \rightsquigarrow l_{i-1}\}$;
- $Args^i_j = \{l_x : Support(N_x) \vdash_{\{\}} \sigma_x | x = 1, \ldots, i-1\}$, for $j = 1, \ldots, k_i, k_i+1, \ldots, k_i+c_i-1$,
  $Args^i_{k_i+c_i} = \{l_x : Support(N_x) \vdash_{\{\}} \sigma_x | x = 1, \ldots, i\}$;
- $Att^i_j = \{l_1 \rightsquigarrow \emptyset\} \cup \{l_x \rightsquigarrow l_y | l_x : \_ \vdash_\_ \sigma_x, l_y : \_ \vdash_\_ \sigma_y \in Args^i_j$, and N_x is a child of N_y in $\mathcal{T}\}$, for $j = 1, \ldots, k_i, k_i+1, \ldots, k_i+c_i$.

As in the proof of Theorem C.1, these sequences can be combined and extended to form X-dispute derivations for GB-choices of parameters. Trivially, by construction, $\mathcal{T}^*$ of the dialectical structure computed by this derivation is the original $\mathcal{T}$. □

C.3. Proof of Theorem 8.3

Let $N_1, \ldots, N_k$, $k \geqslant 1$, be a left-most depth-first order on the nodes of $\mathcal{T}$. Let D_{i+1} be the set of all assumptions in the support of all proponent nodes amongst $N_1, \ldots, N_i$, for $1 < i < k$. Let C_{i+1} be the set of all culprits (namely the assumptions in the support of opponent nodes that the child proponent node attacks) in the support of all opponent nodes amongst $N_1, \ldots, N_i$, for $1 < i < k$. The given $\mathcal{T}$ can be trimmed to some sub-tree $\mathcal{T}^*$ of $\mathcal{T}$, with the same root, such that

- for every proponent node N_i, $i > 1$, in $\mathcal{T}^*$, for every assumption α in the support of the argument labelling N, if $\alpha \in D_i$ then no child of N_i in $\mathcal{T}^*$ is labelled by an argument for $\overline{\alpha}$;

[13] Note that N_{i-1} is necessarily the parent of N_i, and is a proponent node.

- for every proponent node N_i, $i > 1$, in $\mathcal{T}^*$, for every culprit α in the support of the argument labelling N, if $\alpha \in C_i$ then N_i is a leaf in $\mathcal{T}^*$.

We can then apply the same construction as in the proof of Theorem C.2 to obtain a structured X-dispute derivation. The dialectical structure computed by this derivation gives an expanded forest consisting of a $\mathcal{T}'$ with the same trimmed version $\mathcal{T}^*$ as $\mathcal{T}$. Trivially, a is the root of $\mathcal{T}'$ and the argument defence set of $\mathcal{T}'$ is a subset of the argument defence set of $\mathcal{T}$. Also, by soundness of structured X-dispute derivations, $\mathcal{T}'$ is admissible. □

References

[1] J.J. Alferes, P.M. Dung, L.M. Pereira, Scenario semantics of extended logic programs, in: Proc. 2nd International Workshop on Logic Programming and Non-monotonic Reasoning, MIT Press, 1993, pp. 334–348.
[2] P. Besnard, A. Hunter, Elements of Argumentation, MIT Press, 2008.
[3] A. Bondarenko, P. Dung, R. Kowalski, F. Toni, An abstract, argumentation-theoretic approach to default reasoning, Artificial Intelligence 93 (1–2) (1997) 63–101.
[4] D. Bryant, P.J. Krause, An implementation of a lightweight argumentation engine for agent applications, in: Proc. of JELIA06, 2006, pp. 469–472.
[5] D. Bryant, P.J. Krause, G. Vreeswijk, Argue tuProlog: A lightweight argumentation engine for agent applications, in: P.E. Dunne, T.J.M. Bench-Capon (Eds.), Proceedings of the First Computational Models of Argument (COMMA 2006), in: Frontiers in Artificial Intelligence and Applications, vol. 144, IOS Press, 2006, pp. 27–32.
[6] M. Caminada, An algorithm for computing semi-stable semantics, in: K. Mellouli (Ed.), ECSQARU, in: Lecture Notes in Computer Science, vol. 4724, Springer, 2007, pp. 222–234.
[7] C. Cayrol, S. Doutre, J. Mengin, On decision problems related to the preferred semantics for argumentation frameworks, Journal of Logic and Computation 13 (3) (2003) 377–403.
[8] W. Chen, D.S. Warren, Tabled evaluation with delaying for general logic programs, Journal of the ACM 43 (1) (1996) 20–74.
[9] R. Craven, F. Toni, A. Hadad, C. Cadar, M. Williams, Efficient support for medical argumentation, in: T. Eiter, S. McIlraith (Eds.), Proc. 13th International Conference on Principles of Knowledge Representation and Reasoning, 2012, pp. 598–602.
[10] Y. Dimopoulos, B. Nebel, F. Toni, On the computational complexity of assumption-based argumentation for default reasoning, Artificial Intelligence 141 (2002) 57–78.
[11] P. Dung, An argumentation theoretic foundation of logic programming, Journal of Logic Programming 22 (1995) 151–177.
[12] P. Dung, On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games, Artificial Intelligence 77 (1995) 321–357.
[13] P. Dung, R. Kowalski, F. Toni, Dialectic proof procedures for assumption-based, admissible argumentation, Artificial Intelligence 170 (2006) 114–159.
[14] P. Dung, P. Mancarella, F. Toni, Computing ideal sceptical argumentation, Artificial Intelligence 171 (10–15) (2007) 642–674 (Special Issue on Argumentation in Artificial Intelligence).
[15] P.M. Dung, R.A. Kowalski, F. Toni, Assumption-based argumentation, in: I. Rahwan, G. Simari (Eds.), Argumentation in AI: The Book, Springer, 2009, pp. 199–218.
[16] P.M. Dung, P.M. Thang, A unified framework for representation and development of dialectical proof procedures in argumentation, in: C. Boutilier (Ed.), IJCAI, 2009, pp. 746–751.
[17] P.M. Dung, P.M. Thang, F. Toni, Towards argumentation-based contract negotiation, in: Proc. COMMA'08, IOS Press, 2008, pp. 134–146.
[18] P.M. Dung, F. Toni, P. Mancarella, Some design guidelines for practical argumentation systems, in: P. Baroni, F. Cerutti, M. Giacomin, G. Simari (Eds.), Proceedings of the Third International Conference on Computational Models of Argument (COMMA'10), vol. 216, IOS Press, 2010, pp. 183–194.
[19] P. Dunne, T. Bench-Capon, Two party immediate response disputes: properties and efficiency, Artificial Intelligence 149 (2003) 221–250.
[20] P.E. Dunne, The computational complexity of ideal semantics, Artificial Intelligence 173 (18) (2009) 1559–1591.
[21] V. Efstathiou, A. Hunter, Algorithms for generating arguments and counterarguments in propositional logic, International Journal of Approximate Reasoning 52 (6) (2011) 672–704.
[22] U. Egly, S.A. Gaggl, S. Woltran, Aspartix: Implementing argumentation frameworks using answer-set programming, in: ICLP, in: Lecture Notes in Computer Science, vol. 5366, Springer, 2008, pp. 734–738.
[23] K. Eshghi, R. Kowalski, Abduction compared with negation as failure, in: Proc. ICLP, MIT Press, 1989, pp. 234–254.
[24] D. Gaertner, Argumentation and normative reasoning, PhD thesis, Department of Computing, Imperial College, London, 2009.
[25] D. Gaertner, F. Toni, On computing arguments and attacks in assumption-based argumentation, IEEE Intelligent Systems 22 (6) (2007) 24–33 (Special Issue on Argumentation Technology).
[26] D. Gaertner, F. Toni, Hybrid argumentation and its computational properties, in: Proc. COMMA'08, IOS Press, 2008, pp. 183–195.
[27] A.J. Garcia, J. Dix, G.R. Simari, Argument-based logic programming, in: I. Rahwan, G. Simari (Eds.), Argumentation in AI: The Book, Springer, 2009, pp. 153–171.
[28] A.J. García, N.D. Rotstein, G.R. Simari, Dialectical explanations in defeasible argumentation, in: K. Mellouli (Ed.), ECSQARU, in: Lecture Notes in Computer Science, vol. 4724, Springer, 2007, pp. 295–307.
[29] D. Gartner, F. Toni, CaSAPI: a system for credulous and sceptical argumentation, in: G. Simari, P. Torroni (Eds.), Proc. LPNMR Workshop on Argumentation for Non-monotonic Reasoning, 2007, pp. 80–95.
[30] A.V. Gelder, K.A. Ross, J.S. Schlipf, The well-founded semantics for general logic programs, Journal of the ACM 38 (3) (1991) 620–650.
[31] A. Hunter, M. Williams, Argumentation for aggregating clinical evidence, in: 22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010, Vol. 1, Arras, France, 27–29 October, IEEE Computer Society, 2010, pp. 361–368.
[32] H. Jakobovits, D. Vermeir, Dialectic semantics for argumentation frameworks, in: International Conference on Artificial Intelligence and Law, 1999, pp. 53–62.
[33] A. Kakas, P. Mancarella, Stable theories for logic programs, in: Proc. ISLP, MIT Press, 1991, pp. 85–100.
[34] A. Kakas, P. Mancarella, P. Dung, The acceptability semantics for logic programs, in: P.V. Hentenryck (Ed.), Proc. ICLP, MIT Press, 1994, pp. 504–519.
[35] A.C. Kakas, F. Toni, Computing argumentation in logic programming, Journal of Logic and Computation 9 (1999) 515–562.
[36] R.A. Kowalski, A proof procedure using connection graphs, Journal of the ACM 22 (4) (1975) 572–595.
[37] R.A. Kowalski, F. Toni, Abstract argumentation, Journal of Artificial Intelligence 4 (3–4) (1996) 275–296 (Special Issue on Logical Models of Argumentation).
[38] P.-A. Matt, F. Toni, J. Vaccari, Dominant decisions by argumentation agents, argumentation in multi-agent systems, in: P. McBurney, S. Parson, I. Rawan, N. Maudet (Eds.), ArgMAS 2009, 2009, pp. 42–59.
[39] S. Modgil, M. Caminada, Proof theories and algorithms for abstract argumentation frameworks, in: I. Rahwan, G. Simari (Eds.), Argumentation in AI: The Book, Springer, 2009, pp. 105–129.

[40] H. Prakken, An abstract framework for argumentation with structured arguments, Argument and Computation 1 (2010) 93–124.
[41] M. South, G. Vreeswijk, J. Fox, Dungine: A java dung reasoner, in: Proc. COMMA'08, IOS Press, 2008, pp. 360–368.
[42] P.M. Thang, P.M. Dung, N.D. Hung, Towards a common framework for dialectical proof procedures in abstract argumentation, Journal of Logic and Computation 19 (6) (2009) 1071–1109.
[43] F. Toni, Assumption-based argumentation for closed and consistent defeasible reasoning, in: Proc. JURISIN 2007, in Association with The 21th Annual Conference of The Japanese Society for Artificial Intelligence (JSAI2007), 2007, pp. 42–59.
[44] F. Toni, A. Kakas, Computing the acceptability semantics, in: V. Marek, A. Nerode, M. Truszczynski (Eds.), Proc. 3rd International Workshop on Logic Programming and Non-monotonic Reasoning, in: Lecture Notes in Artificial Intelligence, vol. 928, Springer Verlag, 1995, pp. 401–415.
[45] F. Toni, M. Sergot, Argumentation and answer set programming, in: M. Balduccini, T.C. Son (Eds.), Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays in Honor of Michael Gelfond, in: Lecture Notes in Artificial Intelligence, vol. 6565, Springer, 2011, pp. 164–180.
[46] G. Vreeswijk, H. Prakken, Credulous and sceptical argument games for preferred semantics, in: Proc. JELIA, in: Lecture Notes in Computer Science, vol. 1919, Springer Verlag, 2000, pp. 224–238.